VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PROHLÍŽENÍ TEXTOVÝCH SOUBORŮ PRO ALTAP SALAMANDER

VIEWING TEXT FILES IN ALTAP SALAMANDER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN PRYČ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ZBYNĚK KŘIVKA, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

154491

Ústav: Ústav informačních systémů (UIFS)
Student: **Pryč Jan**
Program: Informační technologie
Název: **Prohlížení textových souborů pro Altap Salamander**
Kategorie: Algoritmy a datové struktury
Akademický rok: 2023/24

Zadání:

1. Seznamte se s možnostmi vývoje pro správce souborů Altap Salamander se zaměřením na vývoj pro verzi 4 včetně možnosti využití neoficiálního SDK. Dále se seznamte s možnostmi konfigurace zvýrazňování syntaxe (např. v programu PSPad, KEdit nebo Eclipse) a zobrazení zvýrazňování syntaxe (např. ve zdrojovém kódu).
2. Dle konzultací s vedoucím navrhněte modul pro Altap Salamander pro prohlížení textových formátů s podporou konfigurovatelného zvýrazňování syntaxe pro různé typy textových souborů (např. na základě přípony nebo analýzy souboru). Dle konzultací s vedoucím využijte některý existující formát na popis syntaxe nebo implementujte vlastní, a tím umožněte uživateli zvýrazňování popsat.
3. Návrh implementujte a testujte.
4. Realizaci zhodnoťte a navrhněte budoucí rozvoj a vylepšení.

Literatura:

- Developing for Altap Salamander File Manager. *Altap Salamander* [online]. ALTAP, 2019. Dostupné z: https://www.altap.cz/developers/ [cit. 2020-10-14]
- MORES, Martin. *Prohlížení a porovnávání strukturovaných souborů pro Altap Salamander*. Brno, 2018. Bakalářská práce. VUT FIT, 2018. Dostupné z: https://www.fit.vut.cz/study/thesis/20757/ [cit. 2020-10-14]
- SARKAR, Advait. *The impact of syntax colouring on program comprehension*. In: Proceedings of the 26th Annual Conference of the Psychology of Programming Interest Group: 49-58, 2015. Dostupné z: https://ppig.org/files/2015-PPIG-26th-Sarkar1.pdf [cit. 2020-10-14]

Při obhajobě semestrální části projektu je požadováno:
Body 1, 2 a prototyp k bodu 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Křivka Zbyněk, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 30.10.2023

Abstrakt

Tato bakalářská práce obsahuje analýzu a implementaci pluginu Syntax Highlighting pro Altap Salamander, známý souborový manažer pro operační systémy Windows. Navržený plugin je implementován v C++ a slouží ke zvýrazňování syntaxe ve zdrojových souborech. Cílem je vytvořit nástroj, který dokáže pomoci uživatelům lépe pracovat se zdrojovými soubory a zlepšit efektivitu práce v prostředí Altap Salamander.

Abstract

This bachelor thesis encompasses the analysis and implementation of the Syntax Highlighting plugin for Altap Salamander—a well-known file manager for Windows operating systems. The designed plugin is implemented in C++ and intended for syntax highlighting of source files. The goal is to create a tool that can help users work better with source codes and increases efficiency within the Altap Salamander environment.

Klíčová slova

C++, Altap Salamander, Windows, zvýraznění syntaxe, plugin

Keywords

C++, Altap Salamander, Windows, syntax highlighting, plugin

Citace

PRYČ, Jan. *Prohlížení textových souborů pro Altap Salamander*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zbyněk Křivka, Ph.D.

Prohlížení textových souborů pro Altap Salamander

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zbyňka Křivky. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jan Pryč
7. května 2024

Poděkování

Děkuji Ing. Zbyňkovi Křivkovi, Ph.D. za jeho vstřícný přístup a trpělivost při konzultacích.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Správce souborů patří mezi nezbytné nástroje v počítačovém prostředí a hraje klíčovou roli při manipulaci se soubory a složkami na operačním systému Windows. Altap Salamander je jedním z takových správců souborů, který si získal uznání uživatelů díky své vysoké efektivitě, rychlosti a širokému spektru funkcí. Tento software však není pouhým správcem souborů, poskytuje i prostředí pro rozšíření své funkcionality pomocí pluginů a modulů.

Tato bakalářská práce se zaměřuje na vývoj pluginu pro aplikaci Altap Salamander verze 4, včetně využití neoficiálního Software Development Kit, dále již SDK. Altap Salamander již poskytuje robustní základ pro manipulaci se soubory, nicméně jeho možnosti v oblasti zobrazení zdrojových textových souborů a zvýrazňování syntaxe jsou omezené. Vzhledem k tomu, že práce se zdrojovými soubory, v různých programovacích jazycích, je běžnou činností pro širokou škálu uživatelů, od vývojářů po analytiky, považuji za důležité tyto možnosti rozšířit. Proto jsem se rozhodl vytvořit plugin „SyntaxHighliting“, který se zabývá problematikou konfigurovatelného zvýrazňování syntaxe při prohlížení textových souborů, což je vhodné při práci s programovacími jazyky a textovými formáty.

Kapitola 2 představuje obecný přehled správců souborů. Zaměřuje se na to, co to je Správce souborů, k čemu slouží, jaké funkce tyto nástroje obvykle poskytují, a jaký přínos poskytují svým uživatelům. Jaké typy správců existují a na které platformě se dají využít.

V kapitole 3 této práce se budeme detailněji zabývat Altap Salamanderem jako správcem souborů. Představíme si jeho historii a dále se zaměříme na architekturu tohoto softwaru a jeho modulární povahu, která umožňuje rozšiřování jeho funkcionality.

Kapitola 4 vysvětluje, jaké existují možnosti zvýrazňování a zobrazování zdrojových textových souborů. Tato kapitola zahrnuje popis technologií a metod, které jsou využívány pro zvýraznění syntaxe, a zdůvodňuje výběr konkrétní technologie pro můj plugin.

V kapitole 5 této práce se zaměříme na konkrétní úkol, který spočívá v návrhu, implementaci a testování modulu pro prohlížení textových souborů s podporou konfigurovatelného zvýrazňování syntaxe. Tento modul má za cíl zlepšit uživatelský zážitek a efektivitu při práci s textovými soubory v Altap Salamander. Kapitola 6 se věnuje testování pluginu a zhodnocuje spokojenost uživatelů s pluginem.

Závěr sumarizuje, jak nově vyvinutý plugin pro Altap Salamander přispívá k efektivnější práci se zdrojovými soubory. Diskutujeme o možnostech dalšího rozvoje a o tom, jaký přínos může nový plugin přinést různým skupinám uživatelů.

Celkově bude tato bakalářská práce zkoumat Altap Salamander jako správce souborů, jeho možnosti rozvoje a implementaci nového modulu pro zvýrazňování syntaxe. Bude tak přispívat k hlubšímu porozumění tomuto užitečnému nástroji a jeho schopnosti přizpůsobit se potřebám uživatelů, kteří prácují se zdrojovými soubory.

Kapitola 2

Správci souborů

V této části je vysvětleno, co to je správce souborů a k čemu slouží. Popisuje krátce historii, jak se dá využít a jaké existují typy správců. Který správce se dá použít, se rozhodne podle toho, jaký operační systém[1] uživatel používá, jaké prostředí mu nejvíc vyhovuje a které funkcionality bude chtít využít.

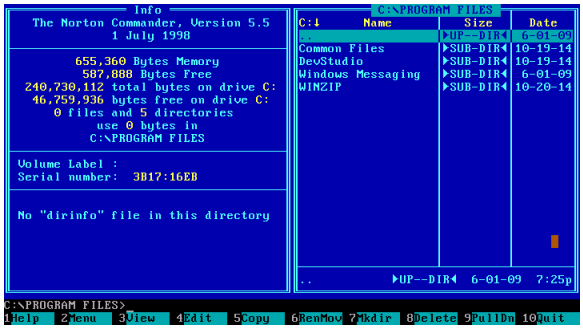
2.1 Historie

Začátek historie správců souborů lze vystopovat až do 70. let 20. století, kdy byly první osobní počítače vybaveny jednoduchými textovými rozhraními. První významný pokrok představovalo vydání operační systému CP/M (Control Program for Microcomputers), Garyho Kildalla v roce 1973, které položilo základy pro pozdější vývoj operačních systémů a správců souborů. V roce 1981 přinesla firma Microsoft operační systém MS-DOS, který se stal standardem pro mnohé počítače té doby. MS-DOS představoval textové rozhraní, kde uživatelé pracovali s příkazy zadávanými z klávesnice. Tento systém byl klíčový pro rané správce souborů, jako byl například Norton Commander, hlavní okno aplikace je na obrázku 2.1. Norton Commander byl uveden na trh v roce 1986. Tento program popularizoval koncept dvoupanelového rozhraní, které umožnilo efektivnější manipulaci se soubory a složkami. [13]

2.2 Definice

Správce souborů je program, který umožňuje uživatelům efektivní manipulaci a organizaci dat na počítači nebo jiném elektronickém zařízení. Jeho hlavní funkce zahrnují prohlížení, vyhledávání, kopírování, mazání, přejmenování a přesouvání souborů a složek. Kromě těchto základních operací mohou správci souborů také nabízet pokročilé funkce, o kterých se píše více v sekci: 2.3 [35]

[1] Software, který ovládá provoz počítače a řídí zpracování programů.[27]



Obrázek 2.1: Hlavní okno aplikace Norton Commander[34]

2.3 Účel

Správa souborů je zásadní pro efektivní a produktivní práci na počítači. Právě díky tomu, že nám umožňuje organizovat a manipulovat data. Většina správců umožňuje uživatelům využívat základní funkce, které jsou popsané v sekci 2.2, avšak někteří umožňují využívat i pokročilejší funkce, jako je třeba porovnání a zipování souborů, prohlížení obrázků, šifrování a dešifrování souborů, pokročilé vyhledávací možnosti, možnosti přizpůsobení uživatelského rozhraní, síťové připojení (např. File Transfer Protocol[2] – FTP klient), integrace se vzdálenými úložišti (Cloudové služby[3]). Tyto a mnoho dalších funkcionalit, vetšina správců již obsahuje, nebo se dájí přidat například pomocí pluginů. Co je to plugin, se píše v části pro Altap Salamander 3.3. [26][33]

2.4 Současný stav

Každý operační systém typicky zahrnuje vlastního správce souborů. Nejčastější, se kterými se může uživatel potkat, jsou File explorer pro Windows, Finder pro MacOs, potom pro Linux existuje více správců, například Dolphin, GNOME Files. V prvních verzích pro Windows byl správce pojmenován File Manager, potom Windows Explorer, poslední název je File Explorer, zkráceně Explorer. Samozřejmě je více správců souborů i pro MacOs a Windows. Konkrétně se zaměříme na Altap Salamander pro Windows, který má vyhrazenou kapitolu 3 sám pro sebe. [14]

Total Commander

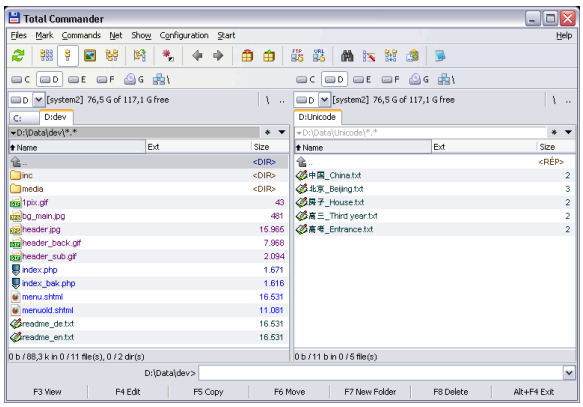
Total Commander je správce souborů pro Windows, který je vysoce přizpůsobitelný a rozšiřitelný prostřednictvím pluginů. Poskytuje řadu funkcí užitečných pro programátory a vývojáře, včetně:

- Dvojpanelové rozhraní
- Podpora více jazyků a Unicode: Tato funkce zajišťuje kompatibilitu s různými jazykovými sadami a umožňuje práci s mezinárodními soubory.

[2] *Síťový protokol pro přenos souborů přes internet*[21]

[3] *Cloudové služby jsou aplikační a infrastrukturní prostředky, které existují na internetu. Poskytovatelé třetích stran uzavírají smlouvy s předplatiteli na tyto služby, což zákazníkům umožňuje využívat výkonné výpočetní zdroje, aniž by museli kupovat nebo udržovat hardware a software.* [16]

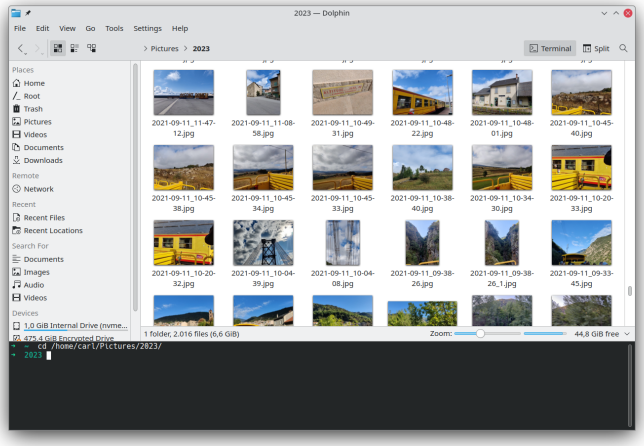
- Pokročilé vyhledávací funkce: Uživatelé mohou efektivně vyhledávat soubory a složky díky pokročilým filtrům a kritériím.
- Pluginy pro archivaci: Podpora pro ZIP, RAR, TAR, GZ, ACE a další formáty.



Obrázek 2.2: Hlavní okno aplikace Total Commander[18]

Dolphin

Dolphin je správce souborů pro uživatele KDE (K Desktop Environment) na platformě Linux. Nabízí integraci s KDE desktopovým prostředím, podporu pro skriptování a pluginy a funkce pro více panelů a záložek. Tyto funkce umožňují uživatelům efektivní organizaci a navigaci v souborech a složkách, stejně jako snadný přístup k souborům na síťových protokolech jako FTP a SFTP.[22]



Obrázek 2.3: Hlavní okno aplikace Dolphin[22]

Porovnání

V této části porovnáme správce souborů Total Commander a Altap Salamander. Oba správce souborů mají společné základní funkce, ale existují rozdíly, které mohou ovlivnit volbu daného správce v závislosti na specifických potřebách.

- **Dvojpanelové rozhraní**: Jak Total Commander, tak Altap Salamander mají dvojpanelové rozhraní, které zjednodušuje souborové operace, jako kopírování a přesouvání mezi dvěma adresáři nebo disky.
- **Podpora pro FTP**: Oba správci, Total Commander i Altap Salamander, nabízejí podporu pro FTP, což umožňuje uživatelům připojení k FTP serverům a správu souborů na dálku. Toto je základní funkce pro mnoho uživatelů pracujících s webovými a síťovými soubory.
- **Podpora archivů**: Total Commander i Altap Salamander umožňují práci s archivovanými soubory, jako jsou ZIP, RAR a další formáty, což zahrnuje extrakci a archivaci souborů přímo z uživatelského rozhraní.
- **Integrace s příkazovým řádkem**: Altap Salamander poskytuje možnosti integrace s příkazovou řádkou, což uživatelům umožňuje rozšířit funkčnost programu pomocí vlastních skriptů nebo příkazů.
- **Správce hesel**: Altap Salamander má integrovaný správce hesel, který umožňuje bezpečně ukládat hesla pro FTP a WinSCP záložky.
- **Podpora pluginů**: Ačkoliv rozsah podpory se může lišit, oba správci umožňují rozšíření funkcionalit skrze pluginy, což uživatelům nabízí možnost přizpůsobit a rozšířit standardní funkce dle individuálních potřeb.

Výběr mezi Total Commanderem a Altap Salamanderem tedy závisí na konkrétních potřebách uživatelů a preferencích v oblasti rozhraní, funkčnosti a rozšiřitelnosti. Většinu funkcí mají společných. [17][5]

Kapitola 3

Altap Salamander

Altap Salamander představuje ztělesnění pružnosti a inovace v kategorii správců souborů. Tato kapitola se zaměřuje na jeho strukturu, historii a vývoj od původní verze po současné vydání. Zaměřuje se na to, jak tento správce souborů poskytuje uživatelům bohatou paletu funkcí a přizpůsobení prostřednictvím pluginů.

3.1 Historie a vývoj

Historie Altap Salamanderu sahá až do devadesátých let minulého století, konkrétně do roku 1997, kdy byl program vyvinut a vydán českými vývojáři. Vznikl díky práci a vizi zakladatelů společnosti Altap s.r.o. Jedním z hlavních zakladatelů je Petr Šolín, který sehrál klíčovou roli v jeho vývoji. Spolu s dalšími členy týmu přivedli Altap Salamander na trh.

První verze byla vydána pod jménem Servant Salamander, byla vyvinuta Petrem Šolínem během jeho studií na Českém vysokém učení technickém v Praze a vydána jako freeware[1] dne 15. 8. 1997. Po ukončení studia, Petr Šolín založil společnost Altap ve spolupráci s Janem Ryšavým a v roce 2001 uvedli první shareware[2] verzi programu. Vývoj pokračoval i v roce 2007. Software byl přejmenován na Altap Salamander a od té doby se projekt rozrostl o mnoho dalších programátorů. Po akvizici společnosti Altap firmou Fine byla vydána verze Altap Salamander 4.0 a v roce 2023 byl projekt uvolněn pod licencí GPL verze 2 jako Open Salamander 5.0. [6]

3.2 Verze

- 15. 8. 1997 - Servant Salamander 1.0 (volně šiřitelné verze)
- 28. 2. 2001 - Servant Salamander 2.0 (šířeno jako shareware)
- 27. 4. 2007 - Altap Salamander 2.5 tato verze přinesla „SDK pro vývojáře pluginů“
- 1. 4. 2014 - Altap Salamander 3.0
- V červnu 2019 oba podílníci společnosti Altap, Jan Ryšavý a Petr Šolín, převedli podíly na majitele firmy FINE[3], která se zaměřuje na vývoj stavebního softwaru. Novými jednateli se tak stali Jiří Laurin a Miloš Vodolan.

[1] svobodný software, který je distribuován bezplatně,
[2] označení pro software chráněný autorským právem, který je možné volně distribuovat
[3] software pro geotechniky

- 11. 6. 2019 - Altap Salamander 4.0

3.3 Architektura

V této sekci je popsána architektura programu Altap Salamander, jazyk ve kterém je program implementován. Rozebírá hlavní okno aplikace, které poskytuje uživatelské rozhraní pro práci se soubory a složkami prostřednictvím dvojpanelového zobrazení. Podrobně se věnuje pluginům, které rozšiřují funkčnost Altap Salamanderu a přizpůsobují aplikaci specifickým potřebám uživatelů.

Modulární architektura

Altap Salamander je postaven na modulární architektuře. Modulární architektura je přístup k návrhu softwaru, který klade důraz na rozdělení systému do menších, samostatných komponent, takzvaných modulů, což zajišťuje, že jakákoliv změna v jednom modulu má minimální dopad na ostatní moduly. Každý modul zabaluje specifickou funkcionalitu a operuje nezávisle. Tento přístup zajišťuje vysokou úroveň flexibility a usnadňuje údržbu softwaru. Lze ho jednoduše vyvíjet a udržovat, aniž by to narušovalo zbytek systému.

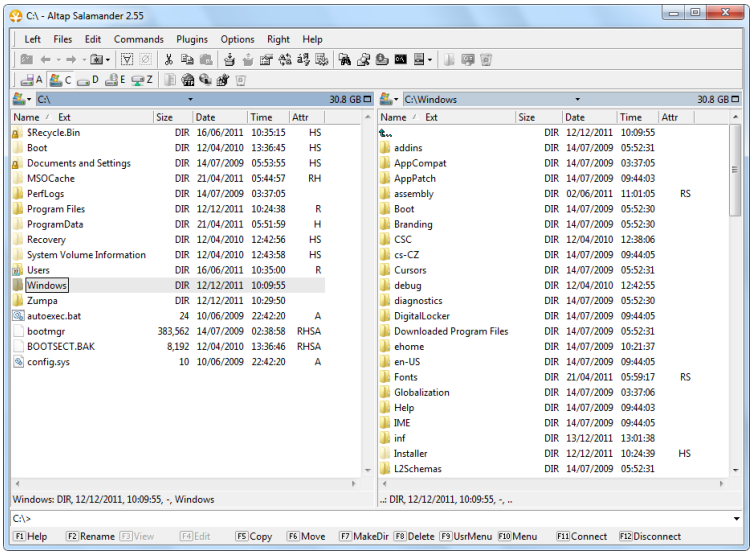
Tyto moduly lze snadno upravovat a rozšiřovat. Implementace modulární architektury pro Altap Salamander, znamená, že vývojáři mohou vytvářet a upravovat pluginy a tím rozšiřovat funkčnost programu, bez nutnosti měnit jeho kód. Aplikace tedy může postupně získávat nové nástroje a funkce, což zvyšuje její hodnotu a užitečnost pro různé typy uživatelů. [25]

Technologie a programovací paradigma

Pro vývoj Altap Salamander byl použit programovací jazyk C++, což je běžné pro desktopové aplikace na Windows. Tento jazyk umožňuje efektivní práci se systémovými zdroji a poskytuje rozsáhlou podporu pro vývoj GUI aplikací. Programovací paradigma zahrnuje objektově orientované programování (OOP), což umožňuje abstrakci a enkapsulaci funkcí do tříd a objektů, což je ideální pro modulární architekturu. [8]

Hlavní okno aplikace

Slouží jako uživatelské rozhraní pro interakci s programem. Obsahuje dva panely, které umožnují uživatelům pracovat se soubory a složkami. Hlavní okno také obsahuje menu, nástrojovou lištu a další prvky pro ovládání programu. Obrázek hlavního okna je vidět zde 3.1.



Obrázek 3.1: Hlavní okno aplikace Altap Salamander[7]

Panely

Panely jsou hlavním prostředkem pro zobrazení obsahu souborů a složek. Altap Salamander podporuje dvojpanelové rozhraní vidno na obrázku 3.1, což znamená, že můžete mít otevřené dva panely současně. Každý panel může zobrazovat obsah různých složek a umožňuje rychle vyhledávat, kopírovat, přesunovat nebo porovnávat soubory.

Pluginy

Altap Salamander umožňuje rozšiřovat svou funkčnost prostřednictvím pluginů. Pluginy jsou skripty, které rozšiřují nebo přidávají nové funkce k již existujícím softwarovým aplikacím. V kontextu správců souborů umožňují pluginy přizpůsobit a rozšířit standardní funkčnost aplikace, aby lépe vyhovovala specifickým potřebám nebo workflow[4] uživatele.

Vytváření vlastních pluginů pro Altap Salamander vyžaduje porozumění API, které aplikace poskytuje, a také znalost programovacího jazyka, ve kterém je Altap Salamander napsán (C++) . Vývojáři musí nejprve definovat funkce, které chce jejich plugin nabídnout, a poté tyto funkce implementovat pomocí tříd a metod, které API poskytuje. Po dokončení vývoje lze plugin snadno integrovat do aplikace, přímo uživatelem prostřednictvím instalačního procesu pluginu.

Integrace nových pluginů do systému Altap Salamander rozšiřuje jeho funkčnost, aniž by docházelo k navýšení velikosti hlavního spustitelného souboru, čímž aplikace udržuje svou původní reaktivitu a efektivitu spuštění. To znamená, že se Altap Salamander spouští pořád stejně rychle, bez ohledu na počet integrovaných funkcí, protože pluginy jsou načítány do paměti pouze v momentě potřeby a po použití jsou z paměti odstraněny. Tento přístup nejenže šetří systémové zdroje, ale také umožňuje uživatelům udržovat aplikaci lehce aktualizovatelnou a flexibilní při přizpůsobování se novým požadavkům. [4]

[4] *Workflow je systém pro správu opakujících se procesů a úkolů, které se odehrávají v určitém pořadí. Jsou to mechanismy, pomocí kterých lidé a podniky realizují svou práci, ať už jde o výrobu produktu, poskytování služby, zpracování informací nebo jakoukoliv jinou činnost generující hodnotu.*[19]

Načítání pluginů do systému je navrženo tak, aby to bylo nejjednodušší jak pro vývojáře, tak pro uživatele. Pluginy jsou typicky rozpoznávány a načítány aplikací automaticky z předem určeného adresáře, což uživatelům umožňuje snadno rozšířit funkčnost Altap Salamander bez potřeby složitého nastavování. Příklady pluginů zahrnují:

- **FTP Plugin**: Plugin umožňující připojení k FTP serverům a práci se soubory na vzdálených serverech.
- **Archivní Plugin**: Plugin pro manipulaci s archivy, který umožňuje vytvářet, extrahovat a spravovat archivní soubory.
- **IEViewer Plugin**: Internet Explorer Viewer Plugin (ieviewer) používá k zobrazení HTML obsahu renderovací engine Internet Exploreru. Toho dosahuje pomocí COM rozhraní, které umožňuje integrovat ovládací prvek prohlížeče Internet Explorer přímo do aplikace.
- **File Comparator**: Umožňuje uživatelům porovnávat 2 soubory zároveň, jak v binární tak i textové podobě. Má několik možností tohoto porovnání, které se dají nastavit v configuraci tohoto pluginu. [3]
- **PictView**: Nástroj na prohlížení obrázků. Podporuje přes 60 různých formátů souborů a vychází z freewarového programu PictView pro DOS a Windows. Díky tomu může být využíván jako efektivní nástroj pro práci s obrázkovými soubory přímo v rámci správce souborů Altap Salamander. [31]

Altap Salamander SDK

Pro vývoj pluginu se dá využít Altap Salamander SDK. SDK obsahuje řadu nástrojů, dokumentaci a ukázkový kód. Tento obsah unadňuje vývoj pluginů pro Altap Salamander.

- **DemoView**: Slouží pro prohlížení textových souborů v integrovaném textovém editoru.
- **DemoMenu**: Tento plugin přidává další položku do menu.
- **DemoPlug**: Nejsložitější plugin z pluginů Demo. Obsahuje skoro všechny možnosti, které by mohl uživatel využít při vytváření nového pluginu.

Vytvořené pluginy mohou být snadno přidány uživatelem pomocí dialogového okna Plugins Manager v Altap Salamanderu. Pokud je potřeba plugin odstranit, lze jej později znovu přidat, což umožňuje snadnou správu a aktualizaci pluginů. Pro vývoj tohoto pluginu bylo využito pluginu Demoplug, což slouží jako kostra programu. Dále je využito pluginu Internet Explorer Viewer (IEViewer). O tom dále v sekci 5. Demoplug plugin má několik důležitých tříd a metod.

Soubor *demoplug.cpp* obsahuje implementaci hlavních funkcí pluginu, včetně jeho inicializace a integrace do hostitelské aplikace.

- `SalamanderPluginEntry()` – Toto je hlavní vstupní bod pro plugin, kde se inicializují základní data pluginu a nastavuje se komunikace s API Salamanderu. Zde také plugin kontroluje kompatibilitu verze Salamanderu a načítá moduly jazykových zdrojů „.slg“.
- `Connect()` – Funkce, která řídí připojení pluginu do hostitelské aplikace.

- `OnConfiguration()` – Otevírá a spravuje konfigurační dialog pluginu.
- `OnAbout()` – Zobrazuje informace o pluginu.
- `Event()` – Zpracovává události, které jsou relevantní pro plugin. Umožňuje pluginu reagovat na změny v prostředí, například změny konfigurace.

Soubor *demoplug.h* deklaruje struktury a funkce použité v *demoplug.cpp*. Jedny s důležitých tříd jsou například:

- `CPluginInterface()` – Třída, která definuje rozhraní pro interakci s API Altap Salamanderu. Obsahuje metody jako *LoadConfiguration()*, *SaveConfiguration()*, a *Configuration()*, které spravují načítání, ukládání a konfiguraci pluginu.
- `CPluginInterfaceForViewer()` – Umožňuje integraci vlastního prohlížeče souborů, jeho zobrazení a kontrolu, zda může plugin daný soubor otevřít.
- `CPluginInterfaceForArchiver()` – Rozšiřuje *CPluginInterface()*, o funkce pro archivování souborů. Jako je přidávání, rozbalování a odstraňování souborů z a do archivu.
- `CPluginInterfaceForFS()` – Obsahuje metody pro práci se souborovým systémem (kopírování, mazání, atd.).

  [9]

Kapitola 4

Možnosti zvýrazňování a zobrazování

V této kapitole jsou zmíněny některé možnosti zvýrazňování syntaxe. Zaměřil jsem se na možnosti zvýrazňování u aplikací, jako jsou VSCode, PSPad a KEdit. Je zde rozebráno zvolené zvýrazňování syntaxe GNU Source-highlight. Nakonec jsou popsány možnosti zobrazení této zvýrazněné syntaxe.

4.1 Zvýrazňování syntaxe

Visual Studio Code

Také známý jako „VS Code“ je vysoce konfigurovatelný editor kódu vyvinutý společností Microsoft, který podporuje mnoho programovacích jazyků a nástrojů. Jednou z funkcí VS Code je zvýraznění syntaxe, které pomáhá vývojářům lépe číst a porozumět kódu tím, že rozlišuje prvky syntaxe různými barvami a styly. Zvýrazňování syntaxe ve VS Code je implementováno pomocí TextMate gramatik.

VS Code umožňuje uživatelům a vývojářům vytvářet vlastní TextMate gramatiky nebo upravovat ty stávající, což poskytuje vysokou flexibilitu. Toto přizpůsobení se provádí skrze rozšíření, které mohou přidat podporu pro nové jazyky nebo změnit způsob zvýrazňování pro stávající jazyky.

Je také možné použít funkci „sémantického zvýraznění“, která přidává základní zvýraznění syntaxe se sémantickým kontextem poskytovaným jazykovým serverem. To umožňuje ještě přesnější zvýraznění založené na analýze kódu, což zlepšuje přehlednost, zejména u složitých projektů. [1]

TextMate

TextMate je pokročilý textový editor, který kombinuje základní funkce textového editoru s pokročilými funkcemi, které usnadňují práci s kódem a textem.

Gramatika TextMate je založena na souborech JSON, které definují, jak zvýraznit různé syntaktické prvky textu. Gramatika obsahuje sadu pravidel založených na regulárních výrazech, která umožňují editorům definovat různé syntaktické prvky, jako jsou klíčová slova, řetězce, komentáře atd.

TextMate Grammar používá koncept „rozsahů“, což jsou identifikátory, které pomáhají klasifikovat tokeny získané z analýzy textu. Každému tokenu je přiřazen rozsah odpovídající

jeho syntaktickému významu. Tokeny lze klasifikovat jako komentáře, klíčová slova nebo řetězce. Poté odeslaná úprava použije motivy, které tyto rozsahy mapují na konkrétní styly a barvy. Tímto způsobem lze vzhled editoru snadno upravovat a přizpůsobovat různým typům kódu nebo osobním preferencím uživatele. [24]

Another Tool for Language Recognition

Another Tool for Language Recognition, česky „Další nástroj pro rozpoznávání jazyka“, dále používaná zkratka ANTLR. Jedná se o pokročilý generátor syntaktických analyzátorů, který se řídí pomocí formálního popisu jazyka, nazývaného gramatika. Tato gramatika je uložena v souboru s příponou „.g4“ a specifikuje seznam pravidel, které popisují strukturu daného jazyka. ANTLR umožňuje generovat analyzátory pro lexikální, syntaktickou analýzu. A analyzátor pro zpracování abstraktních syntaktických stromů. Analyzátory zjišťují, zda řetezce spadají do definovaného jazyka. Umožňuje uživatelům nejen vytvoření vlastních gramatik pro specifické jazyky, ale i efektivní testování a implementaci těchto gramatik. Pro ANTLR není podstatné, o jaký jazyk se jedná.

Proces rozpoznávání v ANTLR začíná lexikální analýzou, kde lexer rozdělí vstupní data na tokeny, které jsou následně zpracovány syntaktickou analýzou, kde parser rozpozná gramatické struktury textu. Výstupem je derivační strom, který poskytuje vizuální reprezentaci struktury textu, na který byla aplikovaná gramatika a je základem pro další zpracování. Pro interakci s derivačními stromy definuje ANTLR komponenty zvané `Listener` a `Visitor`. Tyto komponenty umožňují programátorům reagovat na události, které nastanou během procházení stromu. Toto je obzvláště užitečné pro transformace dat, nebo pro extrakci a další zpracování informací z analyzovaného textu.

ANTLR se tak zaměřuje na návrh vlastních gramatik a vývoj jazykových aplikací, které jsou založeny na generovaných syntaktických analyzátorech. Výsledná flexibilita nástroje ANTLR je klíčová pro uživatele, kteří potřebují efektivně zpracovávat a analyzovat širokou škálu jazyků. [29]

Část souboru s gramatikou pro JSON ve výpise 4.1

```
grammar JSON;
json
    : value EOF
    ;
obj
    : '{' pair (',' pair)* '}'
    | '{' '}'
    ;
pair
    : STRING ':' value
    ;
arr
    : '[' value (',' value)* ']'
    | '[' ']'
    ;
value
    : STRING
    | NUMBER
    | obj
    | arr
    | 'true'
```

```
    | 'false'
    | 'null'
    ;
STRING
    : '"' (ESC | SAFECODEPOINT)* '"'
    ;
NUMBER
    : '-'? INT ('.' [0-9]+)? EXP?
    ;
WS
    : [ \t\n\r]+ -> skip
    ;
```

Výpis 4.1: JSON.g4

Celý soubor uveden zde: [28]

PSPad

PSPad používá pro zvýrazňování syntaxe soubory ve formátu „.ini“. Každý jazyk má svůj vlastní soubor s definicí jazyka. Tento soubor obsahuje definice, které určují, jak se mají zvýrazňovat klíčová slova, komentáře, řetězce a další syntaxe. Tyto definice jsou specifické pro každý programovací jazyk, někdy se však mohou opakovat. Příklad definice v souboru „.ini“ ve výpise 4.2

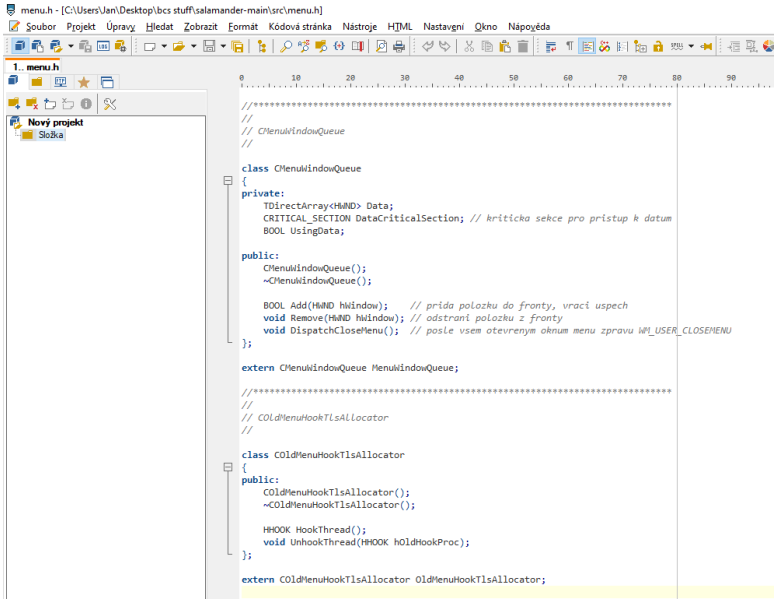
```
; Comments, are ignored
; Bold in brackets are section name
[Settings]
; File name
Name=OtherCad
; List of file extension
FileType=*.prg,*.bpk
; Comments settings (optional) 1 = Yes, 0 = No
ANSIComment=0
PasComment=0
CComment=0
SpecComment=1
BasComment=0
SingleQuote=0
Preprocessors=0
....
; List of keywords and reserved words
; Include the = sign at the end of each word.
[KeyWords]
ABS=
ACOS=
ATN=
BOOLEAN=
[ReservedWords]
BOOLEAN=
INTEGER=
STRING=
```

Výpis 4.2: PSPad .ini soubor [30]

Způsob zobrazení zvýraznění je řízen těmito definicemi a realizován prostřednictvím textového editoru PSPad, který interpretuje tyto definice a aplikuje je na text. To umožňuje uživatelům snadno rozpoznat různé části kódu na základě barevného zvýraznění, což výrazně zlepšuje čitelnost kódu a pomáhá v rychlejší orientaci ve zdrojovém kódu.

PSPad také podporuje uživatelsky definované zvýrazňovače, což umožňuje uživatelům vytvářet vlastní sady pravidel pro zvýrazňování, pokud potřebují podporu pro jazyky, které nejsou standardně zahrnuty, nebo pro jazyky, které si vytvořili. Tato flexibilita činí PSPad oblíbeným nástrojem mezi programátory, kteří pracují s různorodými jazyky a formáty kódů. [30]

Příklad zvýraznění syntaxe v aplikaci PSPad je na obrázku 4.1



Obrázek 4.1: Zvýraznění syntaxe v aplikaci PSPad

KEdit

Pro zvýraznění syntaxe používá KEdit soubory definující pravidla (podobně jako PSPad) pro jednotlivé programovací jazyky, známé jako KEDIT Language Definition (KLD) soubory, mají příponu „.kld“. Tyto soubory určují stejně jako v PSPadu, jaký text bude zobrazen jako komentáře, řetězce, klíčová slova atd., avšak na to používá jiný formát. Ve svém jádru KEdit umožňuje uživatelům definovat vlastní parser pro zvýraznění syntaxe a nahrát příslušný KLD soubor, který obsahuje specifická pravidla pro daný jazyk. Příklad KLD soubor s definicí jazyka je vidět ve výpise 4.3

```
* Experimental KEDITW32 syntax highlighting for *.bat or *.cmd files
* loaded with 'parser BATCH BATCH.KLD' for a 'coloring on BATCH' or
* in a profile for 'autocolor .bat BATCH' and 'autocolor .cmd BATCH'.
* Caveat, ECHO and %FOO:bar=baz% constructs do not work as expected.
* Requires: KEDITW32 on Windows NT       (Frank Ellermann, 2016)
:case
 ignore
:comment
*comment REM to ignore the rest of the batch line (better than nothing)
```

```
 line rem any
:number
 integer
:match
 ( )
:identifier
*identifier is a superset of keywords, e.g., adding / would kill echo/
 [a-zA-Z0-9_]
:keyword
*Windows NT internal CMD.exe commands with extensions and sub-keywords.
 assoc
*break                (legacy ON/OFF or breakpoint)
 cd
 ...
```

Výpis 4.3: KLD soubor pro batch na Windows[15]

Tato barevná schémata jsou definována v KLD souborech, které specifikují, jaké barvy a styly mají být použity pro různé typy syntaxe. [23]

4.2 GNU Source-highlight 3.1.8

O programu

GNU Source-highlight je nástroj, který slouží ke zvýraznění syntaxe zdrojových kódů různých programovacích jazyků, jenž slouží primárně k převedení zdrojového kódu na dokumenty formátované pro čtení. Tento nástroj je neocenitelný pro vývojáře a autory technické dokumentace, kteří chtějí, aby byl jejich kód nejen funkční, ale také vizuálně přístupný a srozumitelný pro další uživatele nebo při revizi kódu. Source-highlight, jakožto produkt GNU projektu, je distribuován jako svobodný software, což umožňuje jeho volné šíření a modifikaci v souladu s GNU General Public License.

Význam Source-highlight vychází z jeho schopnosti nejenom identifikovat syntaktické konstrukce, ale také efektivně přizpůsobit zvýraznění potřebám uživatele, a to díky bohaté sadě předdefinovaných jazykových definic a možnosti vytváření vlastních. Tím poskytuje cenný nástroj pro zlepšení čitelnosti kódu, což je klíčové pro revizi kódu v rámci vývojových týmů a ve vzdělávání v oblasti programování.

Práce s GNU Source-highlight nevyžaduje rozsáhlé programovací dovednosti, což z něj činí uživatelsky přívětivý nástroj vhodný pro široké spektrum uživatelů. Kromě jeho použití v textových editorech a vývojových prostředích lze Source-highlight efektivně začlenit i do pluginů pro různé aplikace, kde slouží k transformaci zdrojového kódu do formátovaných a snadno čitelných formátů, jako jsou HTML a LaTeX. [10]

Základní funkce

Hlavní funkcí GNU Source-highlight je konverze zdrojového kódu do dokumentů, které jsou snadno čitelné a esteticky přijatelné, jako je například HTML. Program zpracovává zdrojový kód a přiděluje různé styly pro různé syntaktické prvky, jako jsou klíčová slova, řetězce, komentáře a další. Tento proces není jen o barvách, lze přidělit i typy písma nebo změnit barvu pozadí, v závislosti na preferencích uživatele. [10]

Architektura

Architektura GNU Source-highlight je rozdělena do několika klíčových součástí:

- **Lexikální analyzátor (Scanner)**: Scanner je zodpovědný za rozeznání syntaktických jednotek (tokenů) v kódu. Pro každý jazyk má Source-highlight definici pravidel, podle kterých Scanner identifikuje tokeny.
- **Syntaktický analyzátor (Parser)**: Po rozpoznání tokenů Scannerem přichází na řadu parser, který interpretuje strukturu a vztahy mezi tokeny podle gramatiky konkrétního jazyka.
- **Definice jazyka**: Pro každý jazyk, který Source-highlight podporuje, existuje soubor pravidel, definující, jaké styly a formáty se mají použít pro různé typy tokenů. Tyto definice lze rozšiřovat a modifikovat.
- **Generátor výstupu**: Na základě analyzovaného a rozpoznaného kódu pak generátor vytvoří výstup ve vybraném formátu (např. HTML, LaTeX) s aplikovanými styly. [11]

Proces zvýrazňování

Základem funkčnosti GNU Source-highlight je jeho schopnost analyzovat a zvýrazňovat syntaxi zdrojového kódu. Tento proces je komplexní a zahrnuje několik kroků od načtení zdrojového textu po jeho finální zobrazení s aplikovaným zvýrazněním. Zde je podrobně rozebráno, jak Source-highlight rozpoznává různé syntaktické struktury a jak je transformuje do výstupu, který je přehledný a vizuálně atraktivní.

Prvním krokem v procesu zvýrazňování je **lexikální analýza**, během které je zdrojový text rozdělen na řetězce tokenů. Tokeny jsou základní stavební kameny programovacího jazyka, jako jsou klíčová slova, identifikátory, čísla, operátory a další. GNU Source-highlight používá k rozpoznání těchto tokenů soubory s definicemi pravidel pro každý podporovaný jazyk. Tyto soubory obsahují regulární výrazy a další instrukce, které umožňují Scanneru přiřadit každému tokenu odpovídající syntaktickou kategorii.

Syntaktická analýza následuje po lexikální analýze a zabývá se interpretací struktury kódu. Pomocí gramatiky daného programovacího jazyka parser rozpozná vztahy mezi tokeny a určuje jejich hierarchii. To umožňuje programu rozlišit, například, kde začíná a končí funkce nebo blok kódu.

Stylování je prováděno tak, že je každý token identifikovaný během lexikální analýzy a je přiřazen ke specifickému stylu definovaném v souborech s pravidly pro zvýraznění. Tyto styly určují: **barvu textu, typ písma, pozadí**. Každý jazyk může obsahovat různé sady stylů, které se následně aplikují.

Generátor výstupu poté tyto styly použije při tvorbě finálního dokumentu. Pokud je výstupem HTML, generátor obalí tokeny odpovídajícími HTML tagy a CSS třídami, které reprezentují definované styly. V případě LaTeXu bude generátor používat příkazy specifické pro toto prostředí, aby dosáhl stejného vizuálního efektu.

Důležitá část a jeden z hlavních důvodů výběru GNU Source-highlight je **adaptivní zvýrazňování**, kvůli schopnosti přizpůsobit se různým jazykům a jejich specifikým vlastnostem, umožňuje uživateli dynamicky načítat specifikace pro zdrojové jazyky a formáty výstupu, což znamená, že je možné jeho schopnosti rozšiřovat bez nutnosti překompilování programu. Tato adaptivita je klíčová při přidávání podpory pro nové programovací jazyky nebo výstupní formáty. [11]

Popis formátu jazyka pro definici

Zde je vysvětleno, jak vytvořit vlastní definici jazyka. Source-highlight používá specifickou syntaxi pro definování prvků zdrojového jazyka (např. klíčová slova, řetězce, komentáře atd.). Prvky jazyka jsou specifikovány v souboru s příponou „.lang“ a načítané dynamicky, prostřednictvím jednoduché syntaxe.

Definice jsou interně použity k vytváření regulárních výrazů, které slouží k zvýrazňování prvků. Používají se regulární výrazy poskytované knihovnou Boost, která se instaluje v sekci 5.5 konkrétně „Problém s boost::regex knihovnou“. Při psaní vlastních definic jazyka je možné se setkat s regulárními výrazy. To se ale ve většině případů nestane. Ale některé definice jazyka mohou vyžadovat použití složitějších regulárních výrazů, například `Perl`.

Zde jsou uvedené příklady, jak definovat klíčová slova, komentáře, symboly a další.

Klíčová slova:

```
keyword = "alfa|beta|gamma" nonsensitive
```

Tímto příkazem se nastavují klíčová slova „alfa“, „beta“ a „gamma“ jako elementy, které budou zvýrazněny. Příkaz „`nonsensitive`“ znamená, že zvýraznění nebude závislé na velikosti písmen. Takže například klíčová slova „Alfa“, „BETA“ a „gamma“, budou všechny zvýrazněny stejně.

Komentáře a symboly:

```
comment start "//"
```

Značí začátek jednořádkového komentáře, tudíž kdekoliv se v kódu objeví „`//`", všechno za tímto znakem až do konce řádku bude považováno za komentář.

Zde je důležité si uvědomit, že pořadí definic jazyka je zásadní, protože se používá při shodování regulárních výrazů. Je nutné se ujistit, že pokud existují definice, které začínají stejnými znaky, tak nejdříve musí být uveden ten nejdelší výraz.

Špatně:

```
symbol = "/"
comment start "//"
```

V tomto případě by znak „/“ byl chybně identifikován jako symbol dříve, než by byl rozpoznán jako začátek komentáře.

Správně:

```
comment start "//"
symbol = "/"
```

Zde je nejdříve definován komentář, takže „`///`" je rozpoznáno jako začátek komentáře, ne jako symbol.

Víceřádkové komentáře:

```
environment comment delim "/**" "*/" multiline begin
```

Tohle nastavení určuje, že cokoliv mezi „`/**`" a „`*/`" bude zpracováváno jako komentář.

Redefinice a substituce :

Tyto dvě funkce jsou užitečné, když je potřeba definovat jazyk s využitím stávající definice jazyka s některými změnami. Při použití redefinice se smažou všechny předchozí

definice daného jazykového prvku novou definicí. Nová definice jazykového prvku bude umístěna přesně v místě nové definice.

Příklad využití redefinice:

```
include "caml.lang"
redef keyword = "abstraction|abstype|and|andalso..."
redef type = "int|byte|boolean|char|long|float|double|short|void"
```

Jelikož se nová definice jazykového prvku objeví přesně v místě redefinice, znamená to, že takový regulární výraz bude shodován pouze tehdy, pokud nebudou shodovány všechny předchozí. To může v některých případech vést k nežádoucím výsledkům.

To znamená, že tento úsek kódu:

```
keyword = "foo"
keyword = "bar"
type = "int"
redef keyword = "myfoo"
```

se interpretuje takto:

```
type = "int"
keyword = "myfoo"
```

Namísto redef, lze použít subst. subst je podobné redef, ale nahrazuje první předchozí definici daného prvku přesně v bodě té první definice.

To znamená, že následující kód:

```
keyword = "foo"
keyword = "bar"
type = "int"
subst keyword = "myfoo"
```

se interpretuje takto:

```
keyword = "myfoo"
type = "int"
```

Tohle je základ pro definici vlastního jazyka. Více informací ohledně definicí jazyka je na webu GNU Source-Highlight.[12]

4.3 Zobrazování zvýrazněné syntaxe

K zobrazení zvýrazněné syntaxe se využívají různé technologie a nástroje, které umožňují implementaci této funkce v různých vývojových prostředích.

HTML formát

Hypertext Markup Language (HTML) je značkovací jazyk, který je strukturován pomocí značek (tagů), které definují různé webové části stránky.

HTML je základem pro stavbu webových stránek, zejména v kontextu webových aplikací a editačních nástrojů. V kontextu našeho pluginu je využíván pro zobrazování zvýrazněné syntaxe. Tato technologie umožňuje efektivně oddělit obsah od vizuální prezentace. Tohle

provádí pomocí značek, které obalují text a určují jeho význam. Používají značky jako nebo a další, které obalují elementy.

Styluje se pomocí Cascading Style Sheets (CSS), společně s HTML se používá k definování vizuálního stylu syntaktických prvků. Například, různé CSS třídy mohou být přiřazeny k různým typům syntaxe, jako jsou klíčová slova, řetězce nebo komentáře, což umožňuje aplikovat specifické barvy, písma a jiné stylové atributy. [20]

Samozřejmě se dá stylovat i bez CSS, dá se nastavit barva a formát různých elementů i v HTML.

Příklad HTML kódu ve výpise 4.4

```
<font color="#009900">short</font> x<font color="#990000">;</font>
<font color="#009900">short</font> y<font color="#990000">;</font>
<font color="#009900">short</font> cx<font color="#990000">;</font>
<font color="#009900">short</font> cy<font color="#990000">;</font>
```

Výpis 4.4: HTML

Rich Text Format

Rich Text Format (RTF) je formát souboru, pro uložení textu, který obsahuje množinu formátovacích příkazů. RTF soubory obsahují jak nastavení formátování, tak samotný text dokumentu, což zahrnuje tučné, kurzívu, velikost písma a nastavení okrajů. Formát je kompatibilní napříč různými operačními systémy a softwarem, udržuje formátování a strukturu dokumentu a má tendenci mít menší velikost souboru ve srovnání s jinými formáty. Na druhou stranu může mít RTF omezení v pokročilém formátování a může být komplikovanější při zpracování velkých souborů s rozsáhlým formátováním nebo grafikou.

Struktura RTF souboru zahrnuje standardní text dokumentu spolu s řadou speciálních RTF příkazů, které určují formátování. Tyto příkazy jsou obklopeny složenými závorkami a začínají zpětným lomítkem. RTF soubor by vypadal jako běžný textový soubor s řadou těchto kontrolních slov a formátovacích značek.

RTF používá kontrolní slova a skupiny pro aplikaci stylů, včetně barvy textu. Barvy jsou definovány v definici barevné tabulky a jsou aplikovány na text pomocí kontrolního slova „\cfX“, kde X je index barvy. [2]

Příklad RTF ve výpise 4.5

```
{\rtf1\ansi\deff0 {\fonttbl {\f0 Courier;}}
{\colortbl;\red0\green0\blue0;\red255\green0\blue0;}
\tx720\tx1440\tx2880\tx5760
    This line is the default color\line
    \cf2
    \tab This line is red and has a tab before it\line
    \cf1
    \page This line is the default color and the first line on page 2
}
```

Výpis 4.5: RTF

V linuxovém prostředí lze v Bash skriptech zvýraznit text různými metodami. Jednou z běžných metod je použití ANSI escape sekvencí, které umožňují aplikovat různé barvy a textové atributy přímo v konzoli.

Výpis 4.6 demonstruje toto použití.

```
#!/bin/bash
echo -e "\e[32m This text is green. \e[0m"
```

Výpis 4.6: Bash

Další metodou je využití nástroje tput, který poskytuje abstrakci pro manipulaci s textovými atributy a umožňuje lepší přenositelnost a čitelnost kódu:

Druhá metoda je ve výpise 4.7.

```
red=$(tput setaf 1)
reset=$(tput sgr0)
echo "${red}T his text is red. ${reset}"
```

Výpis 4.7: Bash

Markdown

Markdown sám o sobě neobsahuje nativní podporu pro barevné zvýraznění syntaxe, ale mnoho nástrojů a platforem, které Markdown renderují, podporuje zvýraznění pomocí rozšíření nebo konverze do HTML. Když je Markdown soubor převeden do HTML, lze použít CSS nebo JavaScript pro zvýraznění.

Příklad Markdown souboru ve výpise 4.8

```
# Title

This text is *Italics* and **BOLD**.

- first item
- second item

[Link](http://www.example.cz)
```

Výpis 4.8: Markdown

LaTeX

V LaTeXu lze aplikovat na text barvu pomocí příkazu `\textcolor{<COLOR>}{<TEXT>}` Samozřejmě je možno zobrazit i zdrojový kód. A to pomocí `\begin{lstlisting}` a uzavřené `\end{lstlisting}`. Uprostřed těchto příkazů je napsaný zdrojový kód a zobrazený.

Příklad kódu v LaTeXu ve výpise s číslem 4.9

```
\begin{document}
\chapter{\LaTeX}
   V kapitole o LaTeXu je popis ...
    \begin{itemize}
       \item prvek
       \item prvek
    \end{itemize}
\end{document}
```

Výpis 4.9: LaTeX

Kapitola 5

Syntax Highlighting – plugin

V této sekci je zdůrazněno, proč je zvýrazňování syntaxe důležité a jaký má dopad na uživatele. Dále je popsáno vývojové prostředí, ve kterém jsem plugin vytvářel, využité metody a třídy, které plugin používá. A nakonec také jakou strukturu je nutné dodržet pro správné fungování pluginu.

5.1 Význam

Psychologický dopad na uživatele

Zvýrazňování syntaxe je dnes běžné v mnoha textových editorech a vývojových prostředích. I když se může zdát, že jeho přínos je pouze estetický, empirické studie potvrzují, že má významný vliv na pochopení zdrojových kódu.

Zvýraznění různých prvků syntaxe může zlepšit čitelnost kódu a umožnit rychlejší navigaci v kódu, což je nezbytné pro efektivní programování. Studie provedená na univerzitě v Cambridge zjistila, že zvýrazňování syntaxe výrazně zkrátilo čas potřebný k dokončení úloh kódu, zejména pro méně zkušené programátory. To naznačuje, že použití zvýraznění syntaxe může snížit kognitivní zátěž uživatelů a zvýšit jejich produktivitu a efektivitu. [32]

Podpora při ladění kódu

Pro efektivní ladění je nezbytná snadná identifikace a oprava chyb v kódu. Zvýraznění syntaxe zde výrazně přispívá k rozlišení syntaktických konstrukcí, čímž usnadňuje lokalizaci chyb. Použitím odlišných barev pro proměnné, datové typy, řetězce a komentáře se zlepšuje srozumitelnost kódu a snižuje se frekvence „mentálních přepínání“, což jsou momenty, kdy programátor musí změnit svůj zaměřený kontext pro pochopení různých částí programu. Redukce těchto přepínání je zdokumentovaná v Sarkarově studii pomocí sledování pohybu očí. Znamená to, že programátoři mohou lépe udržet v paměti informace o stavu programu, což je klíčové pro rychlé a efektivní ladění. Tento proces zlepšuje udržení kontextu celého programu bez nutnosti neustálého prohlížení různých segmentů kódu, což umožňuje programátorům efektivněji a s menším úsilím najít a opravit chyby. [32]

Vliv na programátory různých zkušeností

Ve studii se projevil zajimavý fenomén. I když syntax highlighting významně zvyšuje rychlost porozumění kódu u méně zkušených programátorů, tak tento efekt se s rostoucí zkušeností snižuje. To však neznamená, že zkušení programátoři z tohoto nemají žádný prospěch.

Například, syntax highlighting může poskytnout zkušeným programátorům vizuální ukotvení ve velkých a komplexních kódech, což usnadňuje rychlou orientaci a identifikaci klíčových částí a struktur. Pro nováčky v programování pak může syntaxhighlighting zásadně usnadnit první kroky v učení se programovacím jazykům tím, že jim poskytuje vizuální vodítka, která jim pomáhají pochopit strukturu a logiku kódu. [32]

5.2 Požadavky na plugin

Vývoj pluginu pro zvýrazňování syntaxe v textových editorech, hraje důležitou roli v čitelnosti kódu. Následují klíčové požadavky, které by měl takový plugin splňovat:

Rozpoznání jazyka kódu: Plugin by měl automaticky detekovat jazyk zdrojového kódu, nebo umožnit uživateli manuálně zvolit jazyk, pro který se má zvýrazňování aplikovat. **Podpora více programovacích jazyků**: Plugin by měl podporovat širokou škálu programovacích jazyků, včetně Python, C, C++, JavaScript, Java a dalších. **Nastavitelná témata zvýrazňování**: Je vhodné ne však nutné, aby si uživatel mohl vybrat styl zvýrazňování z různých barevných schémat, nebo si vytvořit vlastní schéma, které nejlépe vyhovuje jejich potřebám a preferencím. **Uživatelská přívětivost**: Instalace a konfigurace pluginu by měly být jednoduché a intuitivní, což umožní uživatelům snadno začít plugin používat bez nutnosti rozsáhlého nastavování. **Podpora kódování**: Plugin musí podporovat různé kódování, aby umožnil správné zobrazení zdrojových kódů napsaných v různých jazycích a s různými speciálními znaky.

5.3 Zvolené technologie

Pro tvorbu pluginu pro Altap Salamander 4.0 jsem se rozhodl využít SDK dostupné na neoficiálním GitHub repozitáři. [9] Protože v době, kdy jsem začal vyvíjet tento plugin, ještě nebyla k dispozici oficiální verze od vývojářů Salamanderu. Přestože oficiální dokumentace Altap Salamanderu doporučuje využít SDK verze 3.08, kvůli snažšímu testování pluginů. Tak jsem využil SDK 4.0. A to proto, že SDK 3.08 nelze jednoduše integrovat do Altap Salamanderu verze 4.0.[8]

Plugin jsem vytvářel ve vývojovém prostředí Microsoft Studio 2022. Sestavení proběhlo v pořádku, i když na oficiálních stránkách doporučují starší verze, protože v nich byly sestaveny pluginy pro starší verze.

GNU Source-highlight 3.1.8 byl zvolen pro jeho schopnost zvýraznění rozmanité škály programovacích jazyků a také protože se do něj dá implementovat vlastní jazyk pro zvýrazňování syntaxe. Jeho flexibilita a rozšiřitelnost jsou ideální pro tento projekt, který vyžaduje detailní kontrolu nad způsobem zvýraznění. Source-highlight také umožňuje snadnou integraci. Více informací o tomto softwaru je zmíněno zde 4.2.

IEViewer byl vybrán kvůli jeho schopnosti zobrazit zvýrazněné HTML, V první fázi vývoje pluginu jsem využil pluginu Demoplug, který jsem využil jako šablonu, což mi pomohlo postavit pevný základ pro další vývoj. Také obsahoval většinu funkcí, které jsem potřeboval pro svůj plugin použít. Dále jsem využil časti z pluginu IEViewer pro zobrazení zvýrazňeného HTML kódu vygenerovaného Source-Highlightem, ve webovém rozhraní.

5.4 Třídy a metody

Zde je seznam důležitých tříd a metod pluginu pro zvýraznění syntaxe:

- `CPluginInterface::Connect()` – Slouží k připojení pluginu k hlavnímu oknu aplikace Altap Salamander.
- `CPluginInterface::Configuration()` – Vytváří okno pro konfiguraci pluginu.
- `CPluginInterfaceForMenuExt::ExecuteMenuItem()` – Metoda, která reaguje na uživatelské akce provedené v menu.
- `CPluginInterfaceForViewer::ViewFile()` – Vytváří nové vlákno, ve kterém se plugin dále zpracovává.
- `CViewerThread::Body()` – Hlavní smyčka, kde se vytváří okno aplikace a zpracovávají se zprávy ve smyčce.
- `CViewerWindow::WindowProc()` – Funkce zpětného volání, která zpracovává zprávy odesílané do okna.
- `CViewerWindow::SyntaxHighlight()` – Funkce pro zvýrazňování syntaxe. Získání cesty k pluginu. Předání souboru programu pro zvýraznění syntaxe.
- `CIEWindow::CreateSite()` – Funkce pro vytvoření webového prohlížeče, ve kterém se zobrazuje HTML soubor .
- `CIEWindow::Navigate()` – Funkce, která zpracovává HTML soubor a vykresluje ho do okna.

5.5 Implementace pluginu Syntax Highlighting

V této části je rozebraná implementace pluginu Syntax Highlighting. Od vstupu pluginu až po finální zobrazení zvýrazněného HTML kódu v prohlížeči, včetně rozboru konfiguračního okna a práce s ním. Konfigurační okno slouží pro vytváření a upravování definic jazyků pro zvýrazňování 5.2.

Činnost pluginu

V následujících sekcích je rozebrána inicializace pluginu společně s jeho konfigurací a zobrazením.

Inicializace

Činnost pluginu začíná propojením pluginu k hlavnímu oknu aplikace pomocí metody `CPluginInterface::Connect()`. Tato metoda slouží jako základní spojnice, kde plugin a aplikace vytvářejí základní komunikační kanál.

Konfigurace

Při nahrání pluginu do aplikace, se automaticky nastaví přípony, které dovoluje plugin otevírat. Bohužel Altap Salamander neumožňuje nahrát do okna pro přípony více než 260 znaků. To značně limituje počet přípon, které může plugin otevírat, i když to aplikace Source-Highlight umožňuje. Proto je nutné zmínit, že uživatel si může tyto přípony změnit, podle vlastních potřeb. Dovolené přípony je možné najít v konfiguračním okně pod tlačítkem „Show“. V konfiguračním okně pluginu se nastavují vlastnosti zvýrazňování. Pokud

je zaškrtávací políčko „Highlighting“ zaškrtnuté, zvýraznění se projeví v okně prohlížeče pluginu. Konfigurační okno obsahuje tlačítko „Show“, toto tlačítko zobrazí nové okno, ve kterém je vypsán soubor, který obsahuje všechny přípony, které plugin dokáže zvýraznit a zobrazit. Obrázek 5.3 zobrazuje obsah „lang.map“. Dále obsahuje jméno a příponu souboru, který je načtený, umístění souboru a obsah souboru.

Pro přidání nové definice vlastního jazyka je nutné zaškrtnout v pravém horním rohu zaškrtávací políčko „Add file to language rule definitions“. Do souboru „lang.map“ se uloží název a přípona vytvořeného souboru, aby mohl program pro zvýraznění syntaxe rozpoznat nově definovaný jazyk, který má nově zvýrazňovat.

Tlačítka „Load file“ a „Save file“ jsou pro uložení a načtení souborů do konfiguračního okna. Tlačítko „OK“ neukládá soubor, pouze předává příznaky zaškrtávacích políček do pluginu.

V případě, že uživatel bude chtít spustit konfigurační okno z menu (Pluginy → Syntax Highlight → Configuration of Syntax Highlighting), nebo pomocí zkratky (Ctrl + Shift + H) tak se vykoná metoda `CPluginInterfaceForMenuExt::ExecuteMenuItem()`. Tato metoda zpracuje danou akci uživatele. Třída `CPathDialog` vytvoří okno konfigurace a zobrazí ho. Toto okno využívá několik funkcí jako například načtení souboru do okna, uložení souboru v okně, nebo zobrazení okna s přípony. Načtení a uložení souborů se provádí po stisknutí tlačítek „Load file a Save file“, zobrazení okna s přípony je umožněno tlačítkem „Show“. Tlačítka se zpracovávají uvnitř metody `CPathDialog::DialogProc()`. Tlačítko „Show“, zobrazí okno s příponami podobně jako se zobrazuje konfigurační okno, používá se třída `CCtrlLangDialog`, tato třída využívá metody `CCtrlLangDialog::DialogProc()`.

Zobrazení

Tlačítko F3 v aplikaci slouží pro zobrazení prohlížeče pluginu, což je nejpodstatnější část pluginu. Než se začně prohlížeč pluginu používat, musí se nejdřív nastavit ve správci pluginů, tento správce je zobrazen na obrázku 5.1. Plugin zobrazuje pouze textové soubory, jejichž přípony jsou definované v souboru „lang.map“. Tyto přípony lze zobrazit v konfiguračním okně pluginu, které je na obrázku 5.2, pomocí tlačítka „Show“.

Po stisku klávesy F3 je volána funkce `CPluginInterfaceForViewer::ViewFile()`, která vytváří nové vlákno pro zpracování prohlížeče (`CViewerThread`), aby nenarušovala chod hlavní aplikace. Zde se plugin rozhoduje, zda otevře interní prohlížeč, nebo otevře prohlížeč html souborů.

Dále volaná metoda `CViewerThread::Body()` inicializuje okno pro zobrazování pluginu a zpracovává příkazy z klávesnice. Uvnitř metody `CViewerThread::Body()`, je vytvořena třída `CViewerWindow`. Tato třída se dědí z třídy `CWindow` a obsahuje důležitou třídu `CIEWindow` s názvem `m_IEViewer`. V těle metody `Body()`, se vytváří okno aplikace, přiřazuje se mu název, volá se metoda `CViewerWindow::SyntaxHiglight()` a nakonec se zpracovávají zprávy ve smyčce.

V případě, že v konfiguraci není nastavené zvýrazňování, tak se zobrazí interní prohlížeč Altap Salamanderu, který volá metoda `SalamanderGeneral->ViewFileInPluginViewer()`. V druhém případě, že v konfiguraci je nastavené zvýrazňování, tak se v hlavním okně vytvoří podokno, které umožňuje zobrazit HTML. `m_IEViewer.CreateSite(HWindow)` je příkaz, který vytváří okno, které umožňuje zobrazit html. Metoda `CreateSite` ve třídě `CIEWindow` zodpovídá za vytvoření „stránky“ pro vkládané (embedding) komponenty Internet Exploreru (IE). Volá metodu `Create()` na instanci `m_Site`, což je objekt typu `CSite`. Tato metoda je zodpovědná za nastavení a inicializaci IE komponenty.

Metoda `Create()` ve třídě `CSite` zajišťuje komplexnější procesy, které zahrnují inicializaci OLE komponent, nastavení a aktivaci IE objektu. Pomocí funkce `CoCreateInstance` se vytvoří instance Internet Exploreru s využitím CLSID 5.1.

```
static CLSID const my_CLSID_WebBrowser =
{0xeab22ac3, 0x30c1, 0x11cf, {0xa7, 0xeb, 0x0, 0x0, 0xc0, 0x5b, 0xae, 0x0b}};
```

Výpis 5.1: CLSID

Dále se získávají rozhraní jako `IID_IWebBrowser`, `IID_IOleObject` a další, které jsou klíčová pro manipulaci a kontrolu nad IE komponentou. Nastavuje se velikost okna pomocí `SetExtent` a aktivuje se objekt v okně pomocí `DoVerb`.

Nakonec připojí události pomocí `ConnectEvents`, což umožní zachytávání a reagování na události vyvolané IE komponentou. V průběhu celé metody jsou kontrolovány potenciální chyby a případné neúspěchy jsou logovány pomocí `TRACE_E` makra.

Toto okno umožňuje uživateli vizualizovat zvýrazněnou syntaxi. Pokud se správně inicializuje okno, zobrazí se do něj textový soubor, který je zvolen. Tohle je provedeno pomocí metody `CViewerWindow::SyntaxHiglight()`.

Metoda `CViewerWindow::SyntaxHiglight()` dále předává jméno souboru `name`, do objektu `m_IEViewer`. pomocí metody `Navigate()`.

Na obrázku 5.4 je vidět samostatné okno prohlížeče, které má zapnutou podporu pro zvýrazňování.

Činnost programu zvýrazňování

Při zahájení procesu zvýrazňování se systém inicializuje nastavením definic jazyků a nastavení formátorů na základě typu vstupního souboru a požadovaného výstupu. Inicializace se provádí v souboru „sourcehighlight.cpp“ v metodě `SourceHighlight::initialize()`, kde se načte definice stylů z konfiguračních souborů určených pro výstupní jazyk (např. HTML, LaTeX) pomocí metody `parse_outlang_def()`. Vytvoří a nastaví se formátovač „FormatterManager“, který spravuje všechny formátovače pro zpracování zdrojových textových souborů. Načtou se styly ze souboru „.css“, pokud je k dispozici uživatelsky definovaný CSS soubor. Pokud není, tak se použije výchozí stylový soubor.

Metoda `SourceHighlight::highlight()` je klíčová pro zpracování vstupního textového souboru. Zpracovává vstupní data řádek po řádku, k tomu využívá funkci`load_line()`. Po načtení každého řádku `SourceFileHighlighter` vyhodnocuje, zda je řádek v zadaném rozsahu pro zvýraznění pomocí `LineRanges` nebo `RegexRanges`. Pokud je řádek určen k zvýraznění, tak ho dále zpracovává metoda `SourceHighlighter::highlightParagraph()` Tato metoda aplikuje pravidla zvýraznění definovaná v `HighlightState`.

Pro každý zpracovaný řádek se výsledný formátovaný text zapisuje do výstupního streamu pomocí `output->output()`. Tato metoda spravuje zápis zvýrazněného textu a dalších prvků, jako jsou předpony řádků nebo čísla řádků, pokud je to zapotřebí. Konec každého řádku je řízen přídavným voláním `output->output()`, které zajišťuje, že každý řádek je správně ukončen a formátován podle potřeb výstupního formátu. Po zpracování všech řádků, je volána metoda `output->writePostDoc(linePrefix)` pro dokončení výstupního dokumentu, což zahrnuje případné přidání závěrečných elementů nebo patičky dokumentu. V případě tohoto pluginu je využit pouze výstup „.html“, kvůli zobrazování pomocí IEVieweru. Zvýrazněný a formátovaný text je uložen do specifikovaného souboru, který plugin zobrazuje.

Struktura

Je nutné dodržet strukturu složek.

- `\src-highlite-rel_3.1.8\src`
- `lang\english.slg`
- `SyntaxHighlight.spl`

Microsoft Visual Studio 2022

Plugin byl vyvíjen s použitím integrovaného vývojového prostředí (IDE) Microsoft Visual Studio 2022, i přes doporučení od vývojářů Altap Salamander SDK využít starší verze. Visual Studio poskytuje rozsáhlé možnosti pro vývoj různých typů softwaru, včetně desktopových aplikací, mobilních aplikací, a webových služeb. Na implementaci tato skutečnost nemá vliv.

V případě ladění pluginu je nutno Visual Studio připojit, k běžícímu procesu (salamand.exe), protože projekt nevytváří žádný proces, který by se dal odladit, ale vytváří se knihovny DLL. Jelikož DLL soubory nelze spustit samostatně, je nutné mít hostitelskou aplikaci, která DLL knihovnu načte. Ladění se provádí v kontextu této aplikace, což umožňuje testování DLL v reálném provozním prostředí. Klíčovým prvkem ladění je schopnost zastavit provádění programu v předem definovaných bodech, takzvaných breakpointech. Breakpointy umožňují zastavit provádění aplikace, analyzovat stav proměnných a tok programu v místech, kde může docházet k chybám, nebo jsou jinak zajímavá. Ladění zahrnuje krokování kódu buď po řádcích (step over) nebo do hloubky funkcí (step into). To umožňuje analyzovat, jak kód reaguje v různých situacích a přesně identifikovat chyby.

Visual Studio poskytuje rozsáhlé možnosti pro konfiguraci a správu projektových vlastností. Umožňuje přizpůsobit a optimalizovat sestavení aplikací podle specifických potřeb. Jedním z klíčových aspektů je schopnost měnit cílovou platformu a architekturu, což zahrnuje možnosti jako x86(32bit) a x64(64bit).

Dalším důležitým nastavením je možnost výběru mezi režimem Debug a Release. Režim Debug je optimalizován pro ladění, nabízí detailní informace o chybách a je méně optimalizován pro výkon. To umožňuje rychleji identifikovat a opravit chyby. Naopak, režim Release je optimalizován pro výkon a efektivitu, což je ideální pro konečné distribuce produktu.

Navíc, Visual Studio podporuje různé standardy jazyků, jako je ISO C++14, ISO C++17 a další. Tohle umožňuje programátorům využívat nejnovější funkce a zlepšení jazyka C++, a tím zvyšuje efektivitu a výkon aplikací.

GNU Source-highlight 3.1.8

V této sekci jsou shrnuty nutné kroky pro instalaci aplikace GNU Source Highlight na systému Windows s použitím MinGW-w64. Poznámka: Tuto část nemusí provádět uživatel pluginu „Syntax Highlighting“.

Úvod

Pro spuštění nástroje GNU Source-highlight_3.1.8 na Windows je nezbytné použít MinGW, nebo jiný kompilátor pro Windows, MinGW obsahuje sadu nástrojů GNU, která umožňuje uživatelům spustit aplikace, jež jsou, nebo byly tradičně dostupné pouze na UNIX systémech.

Instalace a konfigurace MinGW a MSYS2

MSYS2 je doporučené prostředí pro správu balíčků a snadnou instalaci MinGW. Je možné jej stáhnout z oficiálních stránek MSYS2. Instalační program vede uživatele krok za krokem a po instalaci je třeba otevřít MSYS2 konzoli pro další nastavení.

Po prvním spuštění MSYS2 je nutné provést aktualizaci systému pomocí příkazu: „`pacman -Syu`“. Tento příkaz stáhne a nainstaluje nejnovější verze balíčků a MSYS2. Možná bude potřeba zavřít MSYS2 konzoli ručně a otevřít ji znovu, pokud to systém vyžaduje. V tomto případě se do konzole přidá ještě tento příkaz: „`pacman -Su`“, pro dokončení aktualizace. Tyto příkazy zajistí, že všechny součásti systému budou aktuální.

Dalším krokem je instalace MinGW-w64, což je nástrojová sada, která umožňuje vývoj aplikací pro Windows pomocí GCC (GNU Compiler Collection). Instalace se provádí přes MSYS2.

Příkazem „`pacman -S mingw-w64-x86_64-toolchain`“ se nainstaluje MinGW-w64. Tento balíček zahrnuje kompilátory C a C++, které jsou nezbytné pro sestavení GNU Source Highlight.

Konfigurace systémové cesty: Aby bylo možné kompilátory a další nástroje používat přímo z příkazové řádky Windows, je nutné přidat cestu k binárním souborům MinGW-w64 do systémové proměnné PATH. Cesta by mohla vypadat například takto: „`C:\mingw64\bin`“

Příprava a kompilace GNU Source Highlight

Zdrojové soubory GNU Source Highlight lze stáhnout z oficiálního repozitáře GNU. Po stažení je nutné soubory rozbalit do vhodného adresáře.

Před kompilací je třeba projekt nakonfigurovat. To zahrnuje určení cest k závislostem a specifikaci konfiguračních možností.

„`./configure CXXFLAGS="-I/mingw64/include" LDFLAGS="-L/mingw64/lib"`“ Tento příkaz nastaví cesty k hlavičkovým souborům a knihovnám, které mohou být potřebné pro kompilaci. Po úspěšné konfiguraci lze projekt sestavit a nainstalovat pomocí příkazů „make“ a „make install“.

Problém s boost::regex knihovnou na Windows

Chtěl bych zdůraznit, že tento problém nastává u většiny uživatelů, kteří pracují na platformě Windows a používají GNU Source-highlight_3.1.8.

Pokud skript „`./configure`“ zahlásí chybovou hlášku „ERROR! Boost::regex library not installed.“, tak to znamená, že systém nedokáže najít cestu ke knihovně boost::regex, protože není na nainstalovaná, nebo proto, že k ní nemá správnou cestu.

V mém případě nastal stejný problém, neměl jsem tuto knihovnu nainstalovanou a potom jsem musel najít správnou cestu ke knihovně. Na vyřešení jsem musel provést tyto kroky:

První příkaz „`pacman -S bison diffutils`“, slouží pro instalaci nástrojů, které skript „`./configure`“ nedokázal najít, jako jsou `bison` a `diffutils` (který zahrnuje `cmp` a `diff`).

Druhým příkazem „`pacman -S mingw-w64-x86_64-boost`“, se nainstalují chybějící nebo nesprávně nainstalované komponenty Boost.

Další příkaz, který jsem zadal byl „`./configure`
`CXXFLAGS="-I/mingw64/include" LDFLAGS="-L/mingw64/lib"`
`- -with-boost=/mingw64 - -with-boost-libdir=/mingw64/lib`
`- -disable-dependency-tracking`“

Kde `CXXFLAGS="-I/mingw64/include"` nastaví příznaky kompilátoru tak, aby hledal hlavičkové soubory v zadané cestě. Tato cesta je standardním umístěním pro hlavičkové soubory nainstalované pomocí MSYS2. `LDFLAGS="-L/mingw64/lib"` nastaví příznaky linkeru tak, aby hledal knihovny v zadané cestě, což je standardní umístění pro knihovny nainstalované pomocí MSYS2.

Argmuent `-with-boost=/mingw64` říká skriptu `./configure`, kde hledat Boost. Vzhledem k tomu, že Boost byl nainstalován pomocí MSYS2, jeho soubory by měly být dostupné v této cestě. `-with-boost-libdir=/mingw64/lib`.Specifikuje konkrétní složku pro Boost knihovny, což pomáhá `./configure` skriptu najít potřebné Boost knihovny. Tento příkaz se samozřejmě bude lišit podle toho, kde se složka s knihovnou vyskytuje a jakou verzi uživatel používá.

Po instalaci knihovny boost::regex, se dále pokračuje v instalaci.

Před použitím příkazu „`make`“ je nutné ho nainstalovat do MSYS2. To se provede pomocí příkazu „`pacman -S make`“. Další nutné příkazy jsou: „`pacman -S mingw-w64-x86_64-ctags`“ a „`pacman -S doxygen`“ Tyto nástroje jsou nutné pro správné sestavení.

V souboru „`fileutil.h`“ a „`fileutil.cc`“ je nutné odstranit z kódu dynamické specifikace výjimek, v tomto případě „`throw(IOException)`“, které ISO C++17 považuje za zastaralé a nejsou povoleny. Tento problém se dá vyřešit odstraněním těchto specifikací výjimek z kódu.

Části souborů, které bylo nutno přepsat jsou ve výpise 5.2

```
soubory pred upravou:
string readFile(const string \&fileName) throw (IOException);
string readFile(const string \&fileName) throw (IOException) {}

soubory po uprave:
string readFile(const string \&fileName);
string readFile(const string \&fileName) {}
```

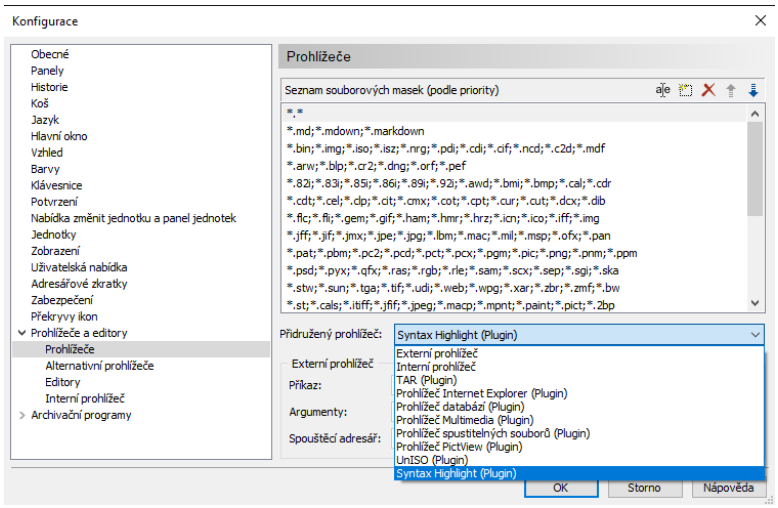
Výpis 5.2: Upravené soubory fileutil.h a fileutil.cc

Před použitím příkazu „`make`“, je potřeba nainstalovat ještě jednu komponentu, a to je „`pacman -S texinfo`“ Tento příkaz nainstaluje Texinfo a jeho nástroje, včetně makeinfo, do systému. Po instalaci by měl být makeinfo dostupný a make by měl být schopen pokračovat v sestavení projektu bez chyb.

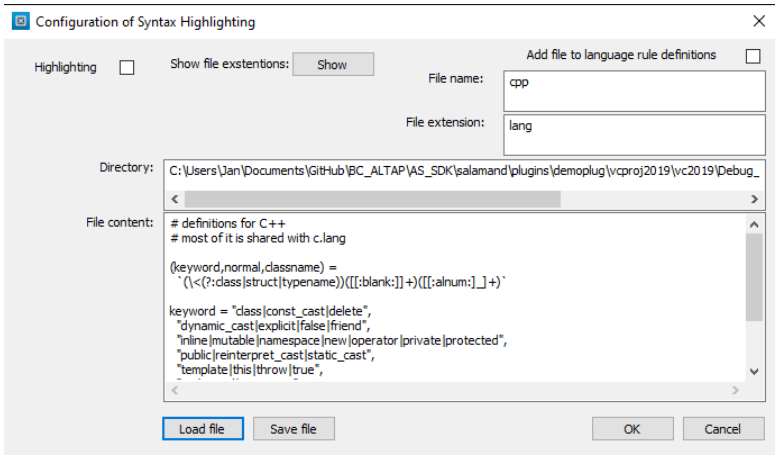
Teď už je potřeba jenom spustit „`make`“ a dokončit instalaci příkazem „`make install`“. Tento seznam příkazů sestaví aplikaci a nainstalují ji do systému. [10]

Usnesení

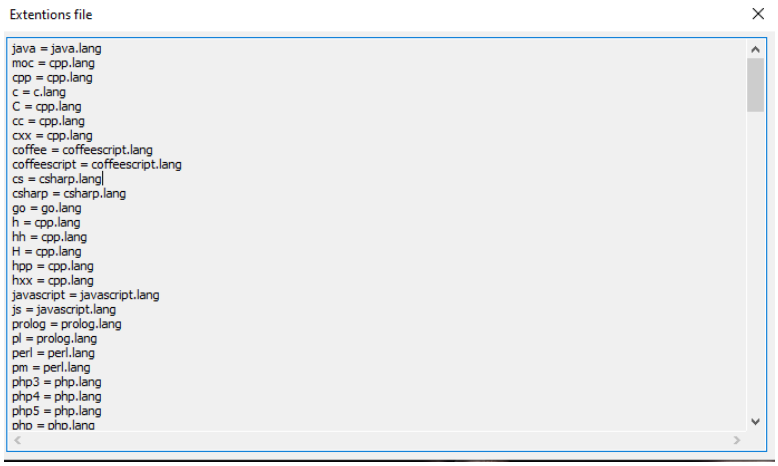
Postup instalace a konfigurace MinGW a MSYS2 na Windows a následná příprava a kompilace GNU Source Highlight, může na první pohled působit složitě, ale poskytuje silnou platformu pro vývoj a distribuci softwaru přímo na Windows, který je tradičně spojen s UNIX systémy. Tento přístup umožňuje uživatelům Windows využívat širokou škálu nástrojů GNU a přináší výhody otevřeného softwaru do jinak uzavřeného systému.



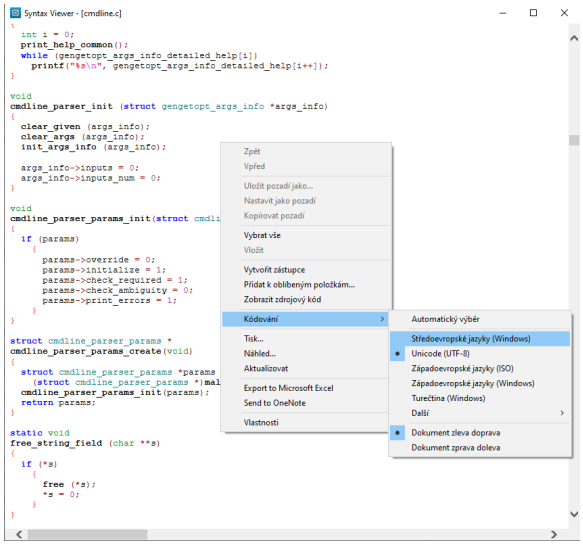
Obrázek 5.1: Okno pro nastavení pluginu Syntax Highlight a přípon souborů, které má prohlížeč pluginu zobrazovat



Obrázek 5.2: Konfigurační okno pro úpravu/vytvoření definic nového jazyka, slouží také pro zapínání/vypínání zvýrazňování syntaxe.
Poznámka: V okně je příklad definice jazyka C++.



Obrázek 5.3: Okno zobrazuje všechny přípony, které plugin dokáže otevřít.



Obrázek 5.4: Plugin Syntax Highlighting s využitím zvýraznění.

Kapitola 6

Testování

V této kapitole je popsána distribuce pluginu na platformě GitHub a fórech Altap Salamanderu. Dále je zde rozebraná zpětná vazba od uživatelů a také to, na co jsem se zaměřil, při kladení otázek ohledně pluginu. Dále je otestována část pluginu, která se zabývá zpracováním vstupního souboru, zvýrazněním dané syntaxe a zobrazením do okna aplikace.

6.1 Distribuce programu

Na fóra Altap Salamanderu jsem vložil odkaz na můj GitHub, na kterém je plugin k dispozici, což umožňuje uživatelům snadný přístup k nejnovějším verzím. Distribuce zahrnuje potřebné soubory GNU-Source-highlight_3.1.8 a samotný plugin. Uživatelé mohou plugin stáhnout jako soubor „.zip“, který se dá lehce extrahovat na Windows, přímo z GitHubu, pod licencí GPL verze 2.

Na fórech Altap Salamanderu se vede diskuze ohledně tohoto pluginu na zvýrazňování.

Zpětná vazba

Vytvořil jsem pro uživatele pluginu dotazník, který zkoumá jejich spokojenost s pluginem. Data jsou sbírána pomocí Google Forms, kde uživatelé hodnotí, jak jim plugin vyhovuje, to znamená – funkčnost, přehlednost a uživatelskou přívětivost. Pod pojmem uživatelská přívětivost je myšleno v tomto případě, jak se jim pracuje s konfiguračním oknem, ve kterém si mohou vytvářet vlastní definice jazyků.

Kritéria, na které jsem se zaměřil při vytváření, jsou:

- Celková spokojenost s pluginem.
- Přehlednost pluginu jako celku.
- Přehlednost konfiguračního okna pluginu.
- Definice vlastních jazyků pro zvýraznění.
- Vzhled pluginu.

Od uživatelů pluginu jsou smíšené ohlasy, ale většina je spíše pozitivních. Největší potenciál na zlepšení je u konfiguračního okna, podle odpovědí uživatelů na obrázku 6.1. Ptal jsem se i na konkrétní podněty, ke zlepšení konfiguračního okna. Odpovědi jsou vidět na obrázku 6.2

6.2 Rychlost pluginu

V této části je popsáno, jak velký vliv má velikost souboru na rychlost zvýraznění syntaxe souboru a načtení do okna aplikace na zobrazení. Testování probíhalo s různě velkými soubory, což umožnilo porovnat výkon pluginu v různých scénářích.

Analýza dat pro různé velikosti souborů ukazuje, že když je zvýraznění syntaxe zapnuto, celková doba zpracování a zobrazení HTML je delší kvůli nutnosti zpracovat celý obsah souboru. V režimu, kdy je zvýraznění vypnuto, aplikace pouze otevře interní prohlížeč a zobrazí obsah, což vysvětluje výrazný rozdíl v rychlosti.

Z grafu v obrázku 6.3, se dají vyvodit zajímavé závěry o výkonnosti pluginu pro zvýrazňování syntaxe ve srovnání s vykreslováním bez zvýraznění.

Přípony testovacích souborů:

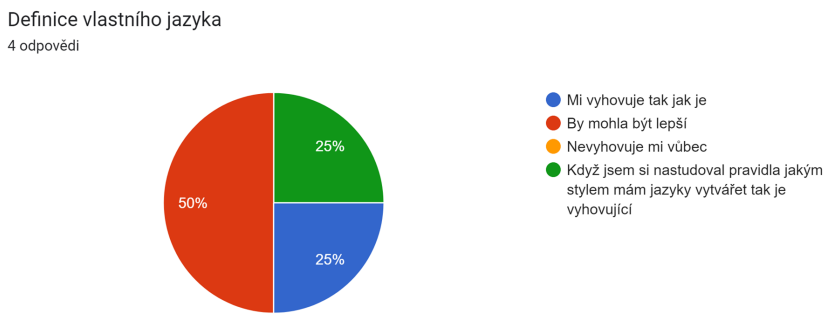
- 6kB – .h
- 20kB – .h
- 50kB – .in
- 80kB – .c

Analýza dat pro :

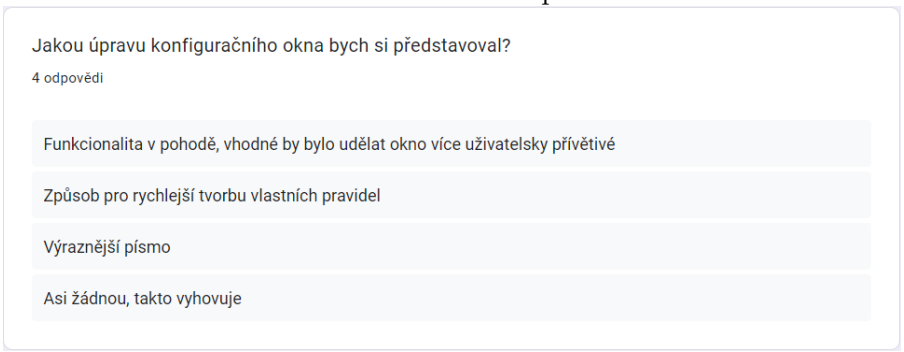
- **Soubory menší než 5kB**:
  Se zvýrazněním syntaxe, dochází ke zvýšení doby zobrazení oproti rychlejšímu zobrazení bez zvýraznění.
- **Soubory 20kB až 50 kB**:
  Když je zvýraznění syntaxe vypnuté, soubory v rozmezí od 20kB do 50kB mají podobný čas vykreslení. Větší rozdíl nastává u zapnuté syntaxe, ale pořád se pohybuje na hranici přijatelnosti.
- **Soubory s velikostí 80kB**:
  S rostoucí velikostí souboru se zvyšuje čas potřebný pro zvýraznění syntaxe, což je indikováno větším nárůstem času u souborů 80kB s zapnutým zvýrazněním ve srovnání s vypnutým. To naznačuje, že režie spojená se zpracováním a formátováním pro zvýraznění se stává významnější s nárůstem velikosti souboru.

Výsledky naznačují, že pro nejmenší soubory je dopad zvýraznění syntaxe na výkon minimální, ale s rostoucí velikostí souboru se stává tento dopad výraznějším. Zvýraznění syntaxe poskytuje detailnější vizualizaci obsahu, ale zároveň vyžaduje více času pro zpracování a zobrazení, což může být nevýhodou při práci s většími soubory.

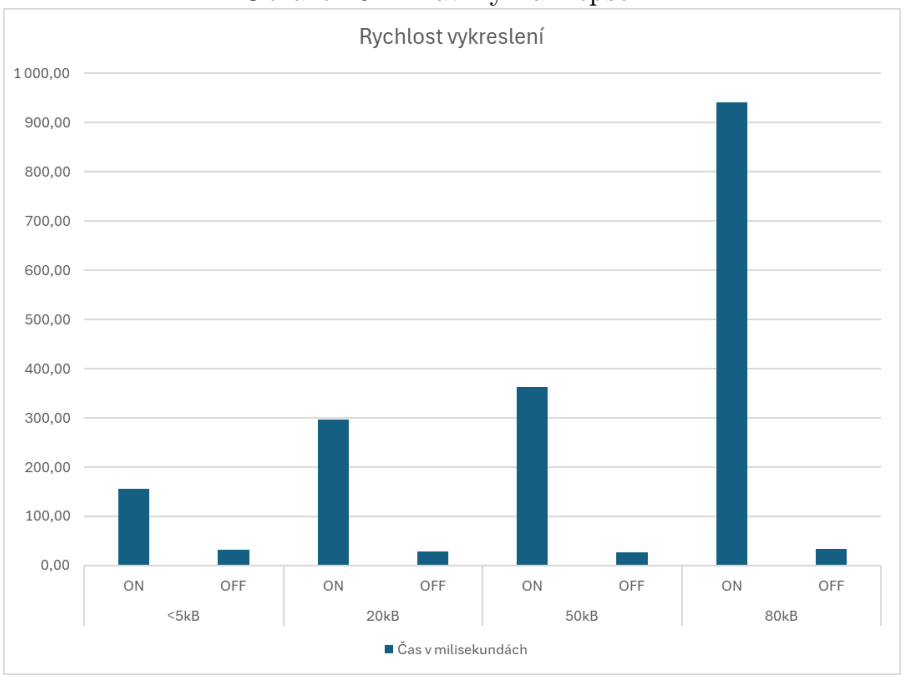
Tento trend poukazuje na důležitost optimalizace pluginu pro zpracování, jak malých tak i velkých souborů, pokud je zvýraznění syntaxe běžně používáno.



Obrázek 6.1: Graf odpovědí.



Obrázek 6.2: Návrhy na zlepšení.



Obrázek 6.3: Graf zpracování různě velikých souborů.

Kapitola 7

Závěr

Tato bakalářská práce se zaměřila na vývoj a implementaci pluginu pro zvýrazňování syntaxe v textových souborech určeného pro správce souborů Altap Salamander. Hlavním cílem této bakalářské práce bylo poskytnout uživatelům tohoto souborového správce nástroj, který by výrazně usnadnil práci se zdrojovými textovými soubory tím, že zlepší jejich přehlednost a čitelnost díky dynamickému zvýraznění syntaxe.

Během práce na této bakalářské práci bylo nezbytné prozkoumat široké spektrum správců souborů a porozumět různým technikám zvýraznění a zobrazení syntaxe. Zvláštní pozornost byla věnována aplikaci Altap Salamander, jejíž funkcionalitu a prostředí bylo nutné detailně pochopit, abych mohl efektivně vyvíjet a integrovat plugin. Za důležité považuji zmínit také program pro zvýrazňování syntaxe, se kterým jsem měl největší úsilí, hlavně při jeho instalaci. Instalace na platformě Windows neprobíhala podle představ, bylo nutné zkusit několik různých překladačů, nakonec se program podařilo sestavit pomocí MINGW-MSYS2. Tato část mi na vývoji zabrala nejvíce času.

Výsledkem této práce je plugin pro Altap Salamander, který rozšiřuje jeho funkčnost o zvýrazňování syntaxe. Plugin je navržen tak, aby podporoval různé programovací jazyky a umožňoval uživatelům si definovat své vlastní, zajišťuje lepší orientaci v kódu, což zvyšuje produktivitu a zjednodušuje práci s kódem.

Od uvedení pluginu do praxe jsem obdržel řadu zpětných vazeb od uživatelů. Tato zpětná vazba byla klíčová pro identifikaci oblastí, ve kterých je možné plugin dále vylepšovat. Uživatelé oceňují jeho funkcionalitu, ale také poukázali na potřebu lepšího uživatelského rozhraní pro definici a správu podporovaných programovacích jazyků.

Na základě těchto zpětných vazeb navrhuji v budoucích verzích pluginu implementovat rozšířené možnosti konfigurace a přizpůsobení, které umožní uživatelům ještě snadněji přidávat a upravovat nastavení pro různé jazyky. Toto vylepšení by mělo zahrnovat intuitivnější grafické rozhraní pro definici pravidel, což by mohlo zásadně zlepšit přívětivost pluginu.

Vývoj tohoto pluginu a získané zkušenosti tak představují významný krok k mému profesnímu růstu a poskytují cenný příspěvek ke komunitě uživatelů Altap Salamander. Tento projekt nejenže zdokonalil mé technické dovednosti v oblasti vývoje software, ale také mi poskytl příležitost přímo spolupracovat s koncovými uživateli a reagovat na jejich potřeby a přání.

Literatura

[1] *Syntax Highlight Guide* [online]. Visual Studio Code [cit. 2024-05-03]. Dostupné z: https://code.visualstudio.com/api/language-extensions/syntax-highlight-guide#textmate-grammars.

[2] Adobe. *What is rich text format and what is it used for?* [online]. Adobe [cit. 2024-10-04]. Dostupné z: https://www.adobe.com/acrobat/hub/what-is-rich-text-format.html.

[3] Altap. *File Comparator* [online]. Altap [cit. 2024-12-02]. Dostupné z: https://www.altap.cz/salamander/features/file-comparator-text-binary-files-compare/.

[4] Altap. *Plugins Concepts* [online]. Altap [cit. 2024-12-02]. Dostupné z: https://www.altap.cz/salamander/help/salamand/plugins_concepts/.

[5] Altap. *Přehled vlastností* [online]. Altap Salamander, 15. srpna 1997 [cit. 2024-26-02]. Dostupné z: https://www.altap.cz/salamander/features/.

[6] Altap. *Seznam změn pro Altap Salamander* [online]. Altap Salamander, 15. srpna 1997 [cit. 2024-26-02]. Dostupné z: https://www.altap.cz/cz/salamander/changelogs/.

[7] Altap. *Altap Salamander - Main Window* [online]. Altap Salamander, 1999 [cit. 2024-26-02]. Dostupné z: https://www.altap.cz/cz/salamander/screenshots/altap-salamander-main-window/.

[8] Altap. *Altap Salamander SDK* [online]. Altap, 2021 [cit. 2024-26-02]. Dostupné z: https://www.altap.cz/salamander/downloads/sdk/.

[9] Altap. *Altap Salamander unofficial SDK 4.0* [online]. Viliam Lejčík, srpen 2021 [cit. 2024-16-12]. Dostupné z: https://github.com/lejcik/as-sdk4-unofficial.

[10] Bettini, L. *GNU Source-highlight 3.1.8* [online]. Free Software Foundation [cit. 2024-13-03]. Dostupné z: https://www.gnu.org/software/src-highlite/.

[11] Bettini, L. *GNU Source-highlight 3.1.8* [online]. Free Software Foundation [cit. 2024-12-04]. Dostupné z: https://www.gnu.org/software/src-highlite/source-highlight.html#How-source_002dhighlight-works.

[12] Bettini, L. *GNU Source-highlight 3.1.8* [online]. Free Software Foundation [cit. 2024-01-05]. Dostupné z: https://www.gnu.org/software/src-highlite/source-highlight.html#Ways-of-specifying-regular-expressions.

[13] Bezroukov, D. N. *DOS history outline* [online]. Softpanorama Society [cit. 2024-01-22]. Dostupné z: https://softpanorama.org/OFM/Paradigm/Ofm_03.shtml.

[14] Dube, R. *What Is a File Manager?* [online]. LifeWire [cit. 2024-26-02]. Dostupné z: https://www.lifewire.com/what-is-a-file-manager-4589189.

[15] Ellermann, F. *KLD file batch for windows* [online]. Frank Ellermann [cit. 2024-16-04]. Dostupné z: https://github.com/frank-e/KLD/blob/master/batch.kld.

[16] Enterprise, H. P. *What are Cloud Services?* [online]. Hewlett Packard Enterprise [cit. 2024-01-26]. Dostupné z: https://www.hpe.com/cz/en/what-is/cloud-services.html.

[17] Ghisler, C. *Total Commander* [online]. Christian Ghisler [cit. 2024-23-02]. Dostupné z: https://www.ghisler.com.

[18] Ghisler, C. *Total Commander Screenshots* [online]. Christian Ghisler [cit. 2024-23-02]. Dostupné z: https://www.ghisler.com/screenshots/en/01.html.

[19] IBM. *What is a workflow?* [online]. IBM [cit. 2024-30-03]. Dostupné z: https://www.ibm.com/topics/workflow.

[20] Janovský, D. *Základy HTML* [online]. jakpsatweb [cit. 2024-10-04]. Dostupné z: https://www.jakpsatweb.cz/zaklady-html.html.

[21] Juviler, J. *FTP Client Essentials* [online]. Jamie Juviler [cit. 2024-01-26]. Dostupné z: https://blog.hubspot.com/website/ftp-client.

[22] KDE. *Dolphin* [online]. KDE [cit. 2024-26-02]. Dostupné z: https://apps.kde.org/dolphin/.

[23] KEdit. *KEDIT Language Definition Files* [online]. [cit. 2024-16-04]. Dostupné z: https://www.kedit.com/doc/KEDIT%20Reference%20Manual.pdf.

[24] MacroMates. *Language Grammars* [online]. MacroMates [cit. 2024-05-03]. Dostupné z: https://macromates.com/manual/en/language_grammars.

[25] Master, A. *Why Use a Modular Architecture in Software Design?* [online]. App Master [cit. 2024-20-03]. Dostupné z: https://appmaster.io/blog/why-use-a-modular-architecture-in-software-design.

[26] McDowell, G. *The 8 Best File Managers for Windows 11/10 in 2022* [online]. Guy McDowell, prosinec 2021 [cit. 2024-26-02]. Dostupné z: https://helpdeskgeek.com/free-tools-review/the-8-best-file-managers-for-windows-11-10-in-2022/.

[27] Merriam Webster.com. *Operating system* [online]. Merriam-Webster [cit. 2024-26-02]. Dostupné z: https://www.merriam-webster.com/dictionary/operating%20system.

[28] Parr, T. *JSON.g4* [online]. Mike Lischke [cit. 2024-05-03]. Dostupné z: https://github.com/antlr/grammars-v4/blob/master/json/JSON.g4.

[29] Parr, T. *The Definitive ANTLR4 Reference*. 2nd. Pragmatic Bookshelf, 2013. ISBN 978-1-934356-99-9. Dostupné z: https://www.ebooks.com/en-cz/book/95960841/the-definitive-antlr-4-reference/terence-parr/.

[30] PSPad. *User highlighters* [online]. [cit. 2024-16-04]. Dostupné z: http://gogogadgetscott.info/pspad/vlastnizvyraznovac.htm.

[31] Ryšavý, J. *Plugins* [online]. Jan Ryšavý [cit. 2024-02-15]. Dostupné z: https://github.com/OpenSalamander/salamander/tree/main/src/plugins.

[32] Sarkar, A. *The impact of syntax colouring on program comprehension* [online]. Psychology of Programming Interest Group [cit. 2024-04-16]. Dostupné z: https://ppig.org/files/2015-PPIG-26th-Sarkar1.pdf.

[33] SCHKN. *Network Manager on Linux with Examples* [online]. devconnected [cit. 2024-26-02]. Dostupné z: https://devconnected.com/network-manager-on-linux-with-examples/.

[34] Win World. *Norton Commander* [online]. Win World [cit. 2024-23-02]. Dostupné z: https://winworldpc.com/product/norton-commander/55x.

[35] Wrike Team. *The basics of file management software* [online]. Wrike Team [cit. 2024-01-22]. Dostupné z: https://www.wrike.com/blog/file-management-software/.

Příloha A

Instalace

V této části je popsáno, jak integrovat plugin do Altap Salamanderu.

A.1 Integrace pluginu

Prvním krokem je získání pluginu. To může uživatel provést stažením z GitHubu, nebo fór altap salamanderu, kde je plugin dostupný. Potom uživatel musí Extrahovat soubory ze složky, do libovolného adresáře, avšak doporučený adresář je: „`C:\Program Files\Altap Salamander\plugins`“. Dále musí uživatel plugin načíst do Altap Salamanderu. `Pluginy` → `Správce pluginů...` → `Přidat...` Zde nahraje plugin s příponou „.spl“. Plugin si automaticky nahraje přípony, které podporuje do konfigurace. Avšak toto okno s přípony je velikostně omezené, takže kdyby uživatel potřeboval změnit nějakou příponu, musí jí přepsat.

Poté stačí v konfiguraci SyntaxHighlightingu zapnout, nebo vypnout zvýrazňování syntaxe. Nebo vytvořit vlastní definici jazyka. A pomocí tlačítka F3 zobrazit prohlížeč pluginu.